Cluster-Based Supervised Classification

by

Huan Wan

A THESIS SUBMITTED TO ULSTER UNIVERSITY
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
IN THE SCHOOL OF COMPUTING
FACULTY OF COMPUTING, ENGINEERING
AND THE BUILT ENVIRONMENT
August 2020

© Huan Wan 2020

All Rights Reserved

DECLARATION

I confirm that the word count of this thesis is less than 100,000 excluding the title page, contents acknowledgements, summary or abstract, abbreviations, footnotes, diagrams, maps, illustrations, tables, appendices, and references or bibliography.

I hereby declare that with effect from the date on which the thesis is deposited in Research Student Administration of Ulster University, I permit:

(i). The Librarian of the University to allow the thesis to be copied in whole or in part without reference to me on the understanding that such authority applies to the provision of single copies made for study purposes or for inclusion within the stock of another library.

(ii). The thesis to be made available through the Ulster Institutional Repository and/or EThOS under the terms of the Ulster eTheses Deposit Agreement which I have signed.

IT IS A CONDITION OF USE OF THIS THESIS THAT ANYONE WHO CONSULTS IT MUST RECOGNISE THAT THE COPYRIGHT RESTS WITH THE AUTHOR AND THAT NO QUOTATION FROM THE THESIS AND NO INFORMATION DERIVED FROM IT MAY BE PUBLISHED UNLESS THE SOURCE IS PROPERLY ACKNOWLEDGED.

Student: Huan Wan

Date: 26/08/2020

ACKNOWLEDGEMENTS

First and foremost, I would like to express my gratitude to my first supervisor, Prof. Hui Wang, for his kind and delicate guidance in my study. His patience and immense knowledge helped me to resolve the countless problems I met throughout the period of research. Without his unconditional support and consistent encouragement, I could not realise this accomplishment. I would also like to thank my other two supervisors, Prof. Bryan Scotney and Dr Jun Liu, for their constant support and encouragement. Their rigorous academic attitude inspires me a lot.

I also want to express my thanks to all my friends who keep me accompany in the UK. The good memories will be always in my heart that we together to attend events organised by the school, take exercises and celebrate various festivals. Besides, I am very happy to have conversations with these friends on both study and life, which make me learn a lot from them and help me overcome my frustrations.

Last but not least, I am grateful to my beloved parents who are always there for me in good and bad times. I also want to express my special thanks to my husband for his constant support and love. He keeps me accompany and takes good care of my life in the UK.

PUBLICATIONS

Journals:

[1]. **Huan Wan**, Hui Wang, Gongde Guo and Xin Wei. "Separability-Oriented Subclass Discriminant Analysis", IEEE Transactions on Pattern Analysis and Machine Intelligence, 2018, Feb 1, vol. 40, no. 2, pp. 409-422.

[2]. **Huan Wan**, Hui Wang, Bryan Scotney, Jun Liu, and Ng W.Y. Wing. "Within-class Multimodal Classification", International Journal of Multimedia Tools and Applications, 2020, 79, no. 39, pp. 29327-29352.

[3]. **Huan Wan**, Hui Wang, Bryan Scotney, Jun Liu, and Xin Wei. "Global Subclass Discriminant Analysis", submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence.

[4]. **Huan Wan**, Hui Wang, Bryan Scotney, Jun Liu, and Xin Wei. "Cluster-based Data Relabelling for Classification", submitted to Pattern Recognition.

[5]. Xin Wei, Hui Wang, Bryan Scotney and **Huan Wan**. "Minimum Margin Loss for Deep Face Recognition", Pattern Recognition, 2020, Jan 1, vol. 97, pp. 107012.

[6]. Xin Wei, Hui Wang, Bryan Scotney and **Huan Wan**. "Selective multi-descriptor fusion for face identification", International Journal of Machine Learning and Cybernetics. 2019: 1-13.

Conferences:

[1]. **Huan Wan**, Hui Wang, Bryan Scotney, and Jun Liu. "A Novel Gaussian Mixture Model for Classification", Proceedings of IEEE International Conference on Systems, Man and Cybernetics (SMC), pp. 3298-3303, 6-9 Oct, 2019, Bari, Italy.

[2]. Xin Wei, Hui Wang, Bryan Scotney and **Huan Wan**. "Precise Adjacent Margin Loss for Deep Face Recognition", Proceedings of IEEE International Conference on Image Processing (ICIP), pp. 3457-3461, 22-25 Sept, 2019, Taipei, Taiwan.

[3]. Xin Wei, Hui Wang, Bryan Scotney and **Huan Wan**. "GicoFace: Global Information-based Cosine Optimal Loss for Deep Face Recognition", Proceedings of IEEE International Conference on Image Processing (ICIP), pp. 3641-3645, 22-25 Sept, 2019, Taipei, Taiwan.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

LIST OF ABBREVIATIONS

(Sort in alphabetical order)

AED	Average Euclidean Distance
AEMA	Average Estimated Mean Accuracy
ALLDA	Adaptive Local Linear Discriminant Analysis
AIC	Akaike Information Criterion
ARFR	Age-related Face Recognition
ASEM	Average Standard Error of the Mean
BBS	Blind Source Separation
BIC	Bayesian Information Criterion
BN	Bayesian Network
CBC	Cluster-based Classification
CBDR	Cluster-based Data Rebelling
CNN	Convolutional Neural Network
CSLBP	Center-Symmetric Local Binary Pattern
DI	Dunn Index
DLDA	Dynamic Linear Discriminant Analysis
DT	Decision Tree
DP	Dirichlet Process
EM	Expectation-Maximization
EMA	Estimated Mean Accuracy
FA	Factor Analysis
GAN	Generative Adversarial Network
GMM	Gaussian Mixture Model
GSDA	Global Subclass Discriminant Analysis
HC-SC	Hierarchical Clustering with Separability Criterion
HMM	Hidden Markov Models
ICA	Independent Component Analysis
IKLDA	Improved Kernel Linear Discriminant Analysis
KDA	Kernel Discriminant Analysis
KNN	K-Nearest Neighbour
KMSDA	Kernel Mixture Subclass Discriminant Analysis
KKT	Karush-Kuhn-Tucker
KSDA	Kernel Subclass Discriminant Analysis
LDA	Linear Discriminant Analysis
LDAC	Linear Discriminant Analysis Classifier

LFW	Labeled Faces in the Wild
LMLP	Linear Multilayer Perceptron
LOOT	Leave-One-Out-Test
LSTM	Long Short-Term Memory
LSVM	Linear Support Vector Machine
MSDA	Mixture Subclass Discriminant Analysis
NB	Naive Bayes
NILM	Non-intrusive Load Monitoring
NLSVM	Nonlinear Support Vector Machine
NMF	Non-negative Matrix Factorization
QDA	Quadratic Discriminant Analysis
PCA	Principal Component Analysis
PRFR	Pose-related Face Recognition
PSVM	Support Vector Machine with polynomial kernel
RRC	Rough-Refine Clustering
RNN	Recurrent Neural Network
RSVM	Support Vector Machine with radial basis kernel
SC-GMM	Separability criterion based Gaussian Mixture Model
SDA	Subclass Discriminant Analysis
SEM	Standard Error of the Mean
SSDA	Separability-oriented Subclass Discriminant Analysis
SVM	Support Vector Machine
SSS	Small Sample Size
TSMLP	Multilayer Perceptron with hyperbolic tangent sigmoid
t-SNE	T-distributed Stochastic Neighbour Embedding
VBGM	Variational Bayesian Gaussian Mixture

ABSTRACT

Supervised classification is one of our fundamental approaches to understanding the world, and is studied in many research areas. Feature extraction and classification learning are two key processes, which significantly influence the performance of supervised classification. Although impressive progress has been made in supervised classification due to the development of feature extraction methods and classifiers, there are still unsolved problems in supervised classification, such as the class imbalance problem and the few-shot classification problem. In this thesis, we focus on the complex boundary problem — it is hard to obtain high classification accuracy for problems with complex decision boundaries due to the existence of subclass structures. We propose a cluster-based approach to supervised classification and develop cluster-based feature extraction methods and cluster-based classification learning methods.

For feature extraction, to find out the importance of considering within-class multimodality for feature extraction, we conduct a study on within-class multimodal data distribution and classification under such a distribution. This study is guided by five important questions about within-class multimodal data. Systematic experiments using a variety of artificial and real data are conducted to answer the five questions, which further lead to some useful findings. In the second study, a new feature extraction method is proposed, called *global subclass discriminant analysis* (GSDA). To extract discriminative features, GSDA first obtains clusters in a global way by clustering the whole data set and derives class-specific clusters based on these global clusters. Then it seeks to maximise interclass distance and minimise intraclass distance based on these class-specific clusters. GSDA is extensively evaluated on a wide range of data through comparison with the

closely related and state-of-the-art feature extraction methods. Experimental results demonstrate GSDA's superiority in terms of accuracy and run time.

For classification learning, in the third study, we propose a *cluster-based data relabelling* (CBDR) method for improving the classification performance of existing classifiers on nonlinear data. CBDR aims to impel classifiers to find cluster-based decision boundaries rather than class-based decision boundaries. Extensive experimentations demonstrate that CBDR dramatically boosts the classification performance of classifiers on nonlinear data, especially for linear classifiers. In the final study, a novel Gaussian mixture model (GMM) classifier is proposed, called *separability criterion based GMM* (SC-GMM) classifier. In SC-GMM, the separability criterion is employed to find the optimal number of Gaussian components for GMM. Experiments have been carried out on various classification tasks. Experimental results demonstrate the superiority of the SC-GMM classifier.

CHAPTER I

Introduction

1.1 Overview of Supervised Classification

Over the past 20 years, the supervised classification problem has drawn great attention from researchers in many fields, such as face recognition [121, 37], human activity recognition [55, 134], text categorisation [4, 60] and disease diagnosis [76, 57]. As one of the essential approaches for object identification and understanding, supervised classification assigns an object into one of a given set of classes based on some given characteristics/attributes/features. For example, supervised classification can be used to determine whether a given face image belongs to *person A* , *person B* or *other persons*, or determine whether a received email is *junk* or *non-junk*. The supervised classification problem can be formulated as: provided a set of data samples $\mathbf{X}$, where $\mathbf{X}$ is usually called the *training data set*, and every sample $\mathbf{x}$ in $\mathbf{X}$ is comprised of a set of features and a class label, the goal is to utilise $\mathbf{X}$ to learn a classification function/decision boundary $f(\mathbf{x})$ so that we can use $f(\mathbf{x})$ to predict the class label of an unseen sample based only on its features. In order to obtain $f(\mathbf{x})$, the classifier is used in the supervised classification process, so the classifier is one of the key processes for achieving high classification accuracy. Apart from the classifier, feature extraction also greatly affects the classification performance. It seeks to remove useless features from $\mathbf{x}$ by creating new features based on combining the given features, since the useful features of a given sample $\mathbf{x}$ are often

overwhelmed by the useless ones [27], resulting in it being difficult for classifiers to learn the optimal classification function. Thus, both classifier and feature extraction make a significant impact on solving supervised classification problems.

1.2 Research Motivation and Contributions

Supervised classification would classify a sample to one or multiple classes. If there is a hierarchical structure among the classes, then being in one class implies another. This type of supervised classification is usually called *hierarchical classification*. For example, animals can be classified into mammals and non-mammals. Mammals can be classified into Euarchonta, Glires, Eulipotyphla, Ferae and others, where Ferae can be further classified into Carnivora and Pholidota. If we know an animal is Carnivora, then we know it is also Ferae as well as mammal. If there is no structure among the classes, then being in one class has no bearing on the sample being in any other class. This type of supervised classification may be called *flat classification*. For example, in human activity recognition, the activity of a person in the given image or video may be classified as walking, running, boxing or other activities, and there is no inherent hierarchy among these activities. So, no further classification will be done after that, which is a typical case of flat classification.

The classification function, $f(\mathbf{x})$, is typically obtained through classification learning. Classification learning is an optimisation process to minimise or maximise an objective function. Typically, flat classification learning is a global optimisation in that the objective is over the whole data. In contrast, hierarchical classification learning is a local optimisation in that there are inherently various objective functions, one for each super-class, with each objective function being calculated over a subset of data that belongs to the super-class, where a super-class is the parent of the class under consideration within the class hierarchy, which can be further divided into different subclasses inheriting the attributes of the super-class. For example, mammals, non-mammals and Ferae are super-classes. Therefore, every super-class has subclass

structures. Subclass structures tend to generate complex decision boundaries between different classes, making it difficult to achieve high classification accuracy [48]. This is called the *complex boundary* problem. Compared with global optimisation, local optimisation contributes to an accurate description of complex decision boundaries, resulting in high classification accuracy. This is consistent with the finding in [43]. Hierarchical classification (local optimisation) algorithms exist [50, 69, 71], but these algorithms tend to be complex, and the class hierarchy is imposed by existing human knowledge through class labels. However, for many classification tasks, class hierarchies are unknown or uncertain [91]. Classification algorithms are mostly for flat classification (global optimisation) even though concepts and things in this world usually have a hierarchical (taxonomic) structure.

In this thesis, we study cluster-based supervised classification learning to solve the *complex boundary* problem of supervised classification. It is a two-level hierarchical classification that learns in a local manner. We do not assume the existence of hierarchical class labels, but we exploit the data structure within each class in the form of class-specific data clusters. We focus on two aspects: feature extraction and classifier design, with the intention to make them widely applicable. Our contributions to feature extraction and classifier design are listed as follows:

- In order to obtain satisfactory feature extraction methods based on discriminant analysis, we deeply study the within-class multimodal data distribution guided by five important questions:

    1) Is it necessary to address within-class multimodality?

    2) How many within-class modalities should we use?

    3) How should we utilise the modalities?

    4) Do we have real benefits from considering these modalities?

    5) If we keep increasing the number of modalities, what will happen?

Through systematic experimentation using artificial and real data, we answer these five questions and obtain some useful findings that are important for the design of new feature extraction methods and the improvement of existing ones.

- A new feature extraction method is developed, which is called *global subclass discriminant analysis* (GSDA). Unlike the existing subclass-based discriminant analysis methods, GSDA selects subclasses in a global way by clustering the whole data set rather than one class at a time. GSDA has been compared with the well-known subclass-based and kernel discriminant analysis methods, and experimental results on the artificial data and real-world data sets demonstrate GSDA's superiority in terms of classification accuracy and run time.

- To enhance the classification performance of the existing classifiers on nonlinear data, in particular for linear classifiers, a *cluster-based data relabelling* (CBDR) method is proposed. The main idea of the CBDR method is to drive classifiers to learn cluster-based decision boundaries instead of class-based ones. Extensive experimentation has shown that CBDR can significantly enhance the classification performance of linear classifiers and even outperform their nonlinear variants on nonlinear data sets. Further experimentation has demonstrated that CBDR can also improve the classification performance of nonlinear classifiers.

- To improve the classification performance of the Gaussian mixture model (GMM) as a classifier, a novel GMM classifier is proposed, which is called SC-GMM classifier. SC-GMM classifier determines the number of Gaussian components for each class based on the separability criterion and takes the number of non-overlapping clusters in every class as the optimal number of Gaussian components of each class. Experimental results show that the SC-GMM classifier outperforms the original GMM classifier and three variants of GMM classifier, and becomes comparable to the commonly used classifiers.

1.3 Outline of the Thesis

This thesis is organised as follows:

- Chapter I provides an overview of the supervised classification problem, our research motivations and contributions, and an outline of the thesis.

- Chapter II first reviews many feature extraction methods that are widely used in supervised classification, including deep feature extraction methods and traditional feature extraction methods. Then, a review of classifiers is provided. Supervised classifiers are generally partitioned into two categories: generative classifiers and discriminative classifiers. Thus, some representative classifiers from these two types are introduced.

- Chapter III presents an extensive study on within-class multimodal data distribution and feature extraction methods. This study is guided by five key research questions. To answer these questions, extensive experiments on a variety of artificial and real-world data sets are presented.

- Chapter IV presents a new feature extraction method. The proposed method is a kind of feature extraction method based on discriminant analysis and attempts to well separate different classes by optimally separating all clusters and concurrently making these clusters as compact as possible. This new feature extraction method is evaluated on a wide range of data and compared with the state-of-the-art algorithms.

- Chapter V develops one generic method for classifiers to improve their classification performance on nonlinear data. The idea of the proposed method is to find global clusters and employ these clusters to relabel data. In this chapter, the proposed method is applied to five existing classifiers, and extensive experiments and comparisons are provided.

- Chapter VI proposes a new Gaussian mixture model (GMM) classifier. This new classifier determines the optimal number of Gaussian components for each class by finding the number of nonoverlapping clusters. In this chapter, the proposed classifier is compared with the original GMM classifier, other variants of GMM and widely used classifiers.

- Chapter VII concludes this thesis and discusses future work.

CHAPTER II

Feature Extraction and Classification Learning

2.1 Introduction

With the rapid development of technology, massive data emerge and grow quickly. These data provide a lot of utilisable information to help us understand the underlying patterns, but it is hard for us to use these data effectively. Useful information/features existing in these data is often overwhelmed by many useless features, such as irrelevant and redundant/repeated features. These useless features not only consume large amounts of storage and computation time, but also hinder training of the classifier that is responsible for obtaining a class label for every datum in the supervised classification problem. So, it is necessary to remove these useless features from an ocean of data. To achieve this, feature extraction is needed.

Typically, in order to obtain satisfactory classification performance, data are usually processed by using a feature extraction method before they are fed into a classifier. Therefore, the feature extraction method and classifier play important roles in supervised classification, and this thesis aims to propose new feature extraction methods and classifiers to achieve high classification accuracy for supervised classification. Before we introduce the proposed new methods, in this chapter some popular and related feature extraction methods and classifiers are overviewed. Firstly, a review of some popular feature extraction methods is presented in Section 2.2, including deep feature extraction

methods and traditional feature extraction methods. Then, the commonly used classifiers in the field of supervised classification are reviewed in Section 2.3.

2.2 Feature Extraction

A feature extraction method is an algorithm that seeks to extract useful features and exclude the irrelevant and repeated features. By doing this, classification accuracy can be greatly enhanced [27]. Many feature extraction methods have been developed, and they are generally divided into two classes: deep feature extraction methods and traditional feature extraction methods. Deep feature extraction methods refer to extracting features through a deep neural network. Usually, we call features extracted in this way as *deep features*. Deep feature extraction methods have become a hotspot in many research areas, in particular for face recognition. In the field of face recognition, convolutional neural network (CNN) is a kind of deep neural network that has been widely used for obtaining deep face features. To extract discriminative features from face images, new deep neural network architectures and loss functions are designed, since they are two primary elements in a deep neural network and significantly influence the quality of deep features. Based on the fact that a deeper network is most likely to produce better deep features, the popular network architectures tend to have a large number of layers: for example, 7-layers AlexNet [63], 16-layers VGGNet [109], 22-layers GoogLeNet [114] and 34-layers ResNet [46]. After the architecture of a deep neural network is fixed, in order to achieve higher accuracy, novel loss functions are designed. The well-known loss functions include Softmax loss [63], Triplet Loss [105], Centre Loss [122] and A-Softmax Loss [77]. Additionally, there are other deep feature extraction methods developed in other applications, including long short-term memory (LSTM) for text classification [75], recurrent neural network (RNN) for human activity recognition [24], and generative adversarial network (GAN) for data generation [41, 128]. Although deep feature extraction methods have caught significant attention recently, they achieve success in only a limited number of applications due to

the following reasons [42]: 1) deep feature extraction methods require a huge amount of training data to obtain promising features, but, in many applications, the enormous amounts of training data required are often hard to obtain; 2) the underlying theory of deep feature extraction methods is not clear, so that it is hard to find suitable values in a theoretical way for the key parameters of deep feature extraction methods, such as the number of layers, the number of nodes in each layer, and the learning rate.

In this thesis, we focus on traditional feature extraction methods. Traditional feature extraction methods refer to algorithms that extract features based on statistical analysis. Due to established theory and strong interpretability, this kind of feature extraction methods has always received attention from different research areas, so that they have wide-ranging applications. The well-known traditional feature extraction methods include principal component analysis (PCA) [124], linear discriminant analysis (LDA) [33, 101], independent component analysis (ICA) [19, 52], non-negative matrix factorization (NMF) [66], and factor analysis (FA) [65]. In the rest of this section, the LDA algorithm will be introduced in detail, since our proposed new feature extraction methods are based on LDA. Additionally, the other two classical and commonly used traditional feature extraction methods, PCA and ICA methods, will be briefly introduced.

2.2.1 Principal Component Analysis

Principal component analysis (PCA) [124] is an important feature extraction method, which studies how to transform the data from a high-dimensional space into a low-dimensional space whilst keeping most of the information of the original features. From a statistical point of view, PCA is also one of the most important multivariate statistical methods. It has been commonly used in various fields such as image processing, pattern recognition, signal processing and data compression.

When using statistical methods to solve multivariate problems, too many variables will increase the computational complexity and the complexity of the analysis. Analysts tend

to use fewer variables and try to keep more information. PCA transforms the original variables into uncorrelated variables and then selects some new variables that are fewer than the original variables but can explain most of the variations in the original data. These new variables are the so-called principal components, with which the task of feature extraction can be achieved.

The basic steps of PCA are summarised as follows:

(i) Given N samples $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N$, where the ith sample $\mathbf{x}_i = (x_i^1, x_i^2, ..., x_i^d)$ has d features, these samples are standardised by:

$$\mathbf{x}_i^{'j} = \frac{\mathbf{x}_i^j - \mu_{\mathbf{x}^j}}{\sigma_{\mathbf{x}^j}}, i = 1, 2, ..., N; j = 1, 2, ..., d, \tag{2.1}$$

where $\mu_{\mathbf{x}^j}$ is the mean of the jth feature column and $\sigma_{\mathbf{x}^j}$ is the standard deviation of the jth feature column.

(ii) After standardisation, the standardised sample matrix $\mathbf{X}$ can be obtained:

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1' \\ \mathbf{x}_2' \\ \vdots \\ \mathbf{x}_N' \end{pmatrix} = \begin{pmatrix} x_1^{'1} & x_1^{'2} & \dots & x_1^{'d} \\ x_2^{'1} & x_2^{'2} & \dots & x_2^{'d} \\ \vdots & \vdots & \ddots & \vdots \\ x_N^{'1} & x_N^{'2} & \dots & x_N^{'d} \end{pmatrix}. \tag{2.2}$$

We then compute the covariance matrix of $\mathbf{X}$ as $\mathbf{R} = \frac{\mathbf{X}^T\mathbf{X}}{N-1}$.

(iii) Eigendecomposition is performed on the covariance matrix $\mathbf{R}$, which gives us m eigenvectors (principal components) $\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_m$ and m eigenvalues $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_m$.

(iv) The eigenvectors are sorted in descending order by the magnitude of their corresponding eigenvalues, and we select the top k principal components.

(v) We then construct the projection matrix $\mathbf{P}$ with the k principal components.

(vi) The matrix $\mathbf{X}$ is thus transformed from d-dimensional feature space into the k-dimensional PCA feature space as follows: $\mathbf{X}_{PCA} = \mathbf{XP}$, where $k \leq d$.

Many variants of PCA have been developed for improving PCA. The most well-known PCA variant is Kernel PCA (KPCA) [87, 59], which requires a kernel trick. Compared with the standard PCA, KPCA has better performance on non-linear data, but it is necessary to select a suitable kernel function and adjust the parameters of the kernel function when using it. Another commonly used PCA variant is Incremental PCA (IPCA) [3][4], which is mainly used to solve the problem of limited RAM. Sometimes, the sample size can be more than one million, and the dimensionality of a sample can be more than one thousand. Fitting the data directly into the RAM is impossible for the standard PCA. Therefore, IPCA is proposed to first divide the data into multiple batches, and then use a partial fitting strategy for each batch recursively, so that the final dimension reduction matrix is obtained. Compared with the standard PCA, IPCA uses much less memory while having a similar performance. Different from the aforementioned PCAs, Cai et al. proposed Sparse PCA [5], which uses L1 normalisation to reduce the influence of the non-principle components. In this way, the process of dimensionality reduction is mainly to reduce the dimensionality of the relatively main components, avoiding the influence of noise on the dimensionality reduction effect. Additionally, in the standard PCA, 2D images are often represented as a 1D feature vector (as a row in an image matrix), which leads to the curse of dimensionality and the loss of image structure information in an image. So, Two Dimensional PCA (2DPCA) [6] was proposed, which allows the 2D images to be represented as a two-dimensional feature vector. Compared with the original PCA, 2DPCA performs better on the 2D image data.

2.2.2 Independent Component Analysis

Based on the statistical characteristics of the input source signal, blind source separation (BSS) is the process of recovering the individual components of the source signal from only the observed signal. Derived from BSS, independent component analysis (ICA) [19] is a multi-dimensional signal processing method, which can find a linear transformation for non-Gaussian data to make the components relatively independent of each other. ICA aims to extract the independent features from data, where the independent features can directly reflect the essential properties of the object being studied. With these features as the input, the classifier can improve the classification speed and accuracy. ICA has a wide range of applications in many fields, such as feature extraction, image processing, and speech processing.

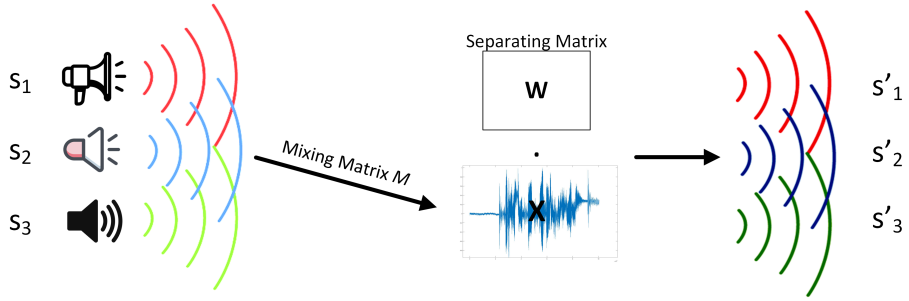


Figure 2.1: The basic idea of ICA.

Figure 2.1 illustrates the basic idea of ICA, where the observed sample matrix $\mathbf{X}$ is the combination of the truth source signal vector $\mathbf{s}$ ($\mathbf{s} = (s_1, s_2, s_3)$) and mixing matrix $\mathbf{M}$. The goal of ICA is to obtain $\mathbf{s}$ based only on $\mathbf{X}$. To extract the source signal from $\mathbf{X}$, ICA seeks to find a separating matrix $\mathbf{W}$. Then, the found source signal $\mathbf{s}'$ ($\mathbf{s}' = (s'_1, s'_2, s'_3)$) that is closely similar to $\mathbf{s}$ is obtained through $\mathbf{W}$.

Therefore, the ICA problem can be formulated as: suppose there are d unknown source signals $s_1, s_2, ..., s_d$, then the observed sample matrix $\mathbf{X}$ can be represented by

$$\mathbf{X} = \mathbf{Ms} = \sum_{i=1}^{d} s_i \alpha_i, \tag{2.3}$$

where $\mathbf{M} = (\alpha_1, \alpha_2, ..., \alpha_d)^T$ is the unknown mixing matrix and $\mathbf{s} = (s_1, s_2, ..., s_d)$ is an unknown *d*-dimensional source vector. In ICA, source signals $s_1, s_2, ..., s_d$ are assumed to be statistically independent, and the dimensionality of $\mathbf{X}$ is assumed to be larger than or equal to d to make the mixing matrix $\mathbf{M}$ have full rank.

Since the source signals are statistically independent, standard linear ICA estimates a separating matrix $\mathbf{W}$ such that $\mathbf{s}' = \mathbf{WX}$ becomes a vector with mutually independent components. Based on different ways of estimating the separating matrix, many ICA algorithms have been proposed and can be roughly divided into two categories: complete/undercomplete ICA and overcomplete ICA. In the case of complete/undercomplete ICA algorithms, they require that the number of features of samples is equal to or greater than the number of sources. The well-known complete/undercomplete ICA algorithms include Fast ICA [51] and JADE (Joint approximation diagonalization of eigenmatrices) [15]. To obtain the separating matrix, Fast ICA aims to minimise the mutual information of the estimated components. It uses the maximum entropy principle to approximate the negative entropy and achieves optimal performance through an appropriate nonlinear function. Compared with the standard ICA, Fast ICA converges faster and the optimisation process is more robust. Different from Fast ICA, JADE aims to extract independent non-Gaussian signals from a data matrix by constructing the fourth-order cumulants from the original data matrix and minimising the sum-of-squares of the off-diagonal elements corresponding to the fourth-order cumulants between the different signals. JADE equates the objective function maximization problem to the joint diagonalization problem of the characteristic matrix of a set of fourth-order cumulant matrices. Compared with the standard ICA, JADE greatly simplifies the computational complexity and also effectively improves separation performance. In the case of overcomplete ICA algorithms, the number of features of samples is fewer than the number of sources, hence samples can't be represented by a unique combination of sources. To overcome the problem, Michael and Terrence proposed to learn overcomplete

representations by using a probabilistic model of observed samples and maximising the probability of each sample based on the probabilistic model [67]. In addition, a semidefinite programming relaxation has been introduced in [99] for overcomplete analysis. Other overcomplete ICA algorithms include overcomplete ICA based on sparse representations [116] and Fourth-Order-Only Blind Identification (FOOBI) algorithm [22].

2.2.3 Linear Discriminant Analysis

Linear discriminant analysis (LDA) is a classic method for discriminant analysis that has been widely used in pattern recognition [68, 70] and machine learning [56, 120]. LDA was originally proposed by Fisher for binary classification in [33] and [34]. It was generalised by Rao [101] for multiclass classification. The basic idea of LDA is to find a transformation/projection matrix $\mathbf{W}$ that projects data into a new space, LDA space, that is spanned by LDA features (or LDA dimensions), such that the interclass distance is maximised and simultaneously the intraclass distance is minimised.

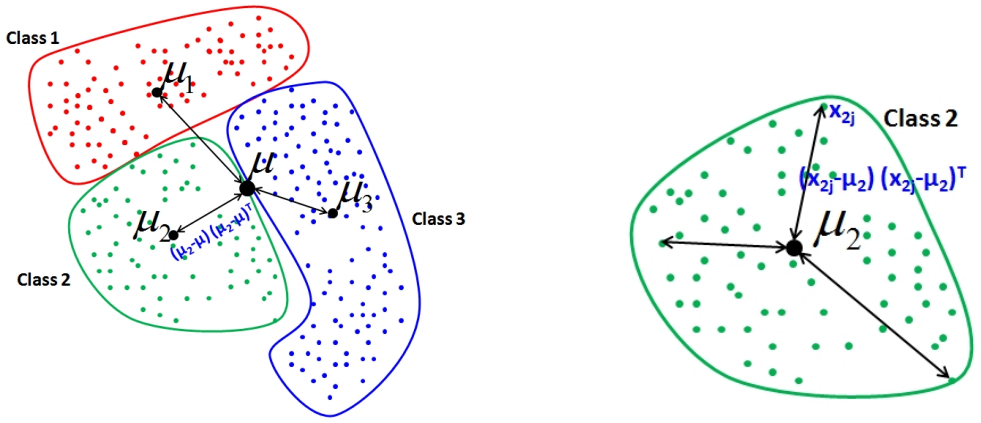
In the LDA algorithm, under the assumption that every class has a normal distribution and has the same covariance matrix, LDA uses a between-class scatter matrix $\mathbf{S_b}$ to measure the interclass distance, and uses a within-class scatter matrix $\mathbf{S_w}$ to measure the intraclass distance. $\mathbf{S_b}$ and $\mathbf{S_w}$ are defined, respectively, as follows:

$$\mathbf{S_b} = \frac{1}{N}\sum_{i=1}^{C} N_i(\mu_i - \mu)(\mu_i - \mu)^T, \tag{2.4}$$

$$\mathbf{S_w} = \frac{1}{N}\sum_{i=1}^{C}\sum_{j=1}^{N_i}(\mathbf{x}_{ij} - \mu_i)(\mathbf{x}_{ij} - \mu_i)^T, \tag{2.5}$$

where N is the number of training samples, N_i is the number of samples in class i, C is the number of classes, μ_i is the mean of class i, μ is the mean of the whole training samples, and $\mathbf{x}_{ij}$ denotes the jth sample in class i. According to Equations (2.4) and (2.5),

we note that $\mathbf{S_b}$ employs the distance between the means of classes and the global mean as the interclass distance (see Figure. 2.2(a)), and $\mathbf{S_w}$ utilises the distance between samples and the mean of their corresponding classes to measure the intraclass distance (see Figure. 2.2(b)).



(a) $\mathbf{S_b}$ (b) $\mathbf{S_w}$

Figure 2.2: Visualisation of how the between-class scatter matrix $\mathbf{S_b}$ and the within-class scatter matrix $\mathbf{S_w}$ are calculated in the LDA algorithm, where differently coloured dots represent samples from different classes.

LDA is an optimisation process, with the following *Fisher objective* [34]:

$$J^{LDA}(\mathbf{W}) = \frac{tr(\mathbf{W}^T\mathbf{S_b}\mathbf{W})}{tr(\mathbf{W}^T\mathbf{S_w}\mathbf{W})}, \tag{2.6}$$

where $tr()$ denotes the trace of a matrix, and $\mathbf{W}$ is a projection matrix that projects data from the data space to the LDA space. In order to find an LDA space that can separate different classes well, LDA needs to find the optimal projection matrix $\mathbf{W}^* = \arg\max_{\mathbf{W}} J^{LDA}(\mathbf{W})$. To obtain $\mathbf{W}^*$, we assume that $F(\mathbf{W}) = \mathbf{W}^T\mathbf{S_b}\mathbf{W}$ and $G(\mathbf{W}) = \mathbf{W}^T\mathbf{S_w}\mathbf{W} - \alpha = 0$, $\alpha > 0$ is a constant. Then the optimisation of LDA is equivalent to finding a projection matrix $\mathbf{W}$ to maximise $F(\mathbf{W})$ under the $G(\mathbf{W})$ constraint. For this we employ a Lagrangian function to obtain $\mathbf{W}^*$. Thus, we define $L = F(\mathbf{W}) - \lambda G(\mathbf{W})$, where $\lambda \neq 0$ is a Lagrange multiplier. By setting the derivative of

$L(\mathbf{W})$ with respect to $\mathbf{W}$ to zero, we get

$$\begin{aligned}\frac{\partial L(\mathbf{W})}{\partial \mathbf{W}} = \frac{\partial \mathbf{W}^T \mathbf{S_b} \mathbf{W} - \lambda(\mathbf{W}^T \mathbf{S_w} \mathbf{W} - \alpha)}{\partial \mathbf{W}} &= 0 \\ \Rightarrow 2\mathbf{S_b}\mathbf{W} - 2\lambda\mathbf{W} &= 0 \\ \Rightarrow \mathbf{S_b}\mathbf{W} &= \lambda \mathbf{S_w}\mathbf{W}.\end{aligned}$$

If $\mathbf{S_w}$ is nonsingular, we can obtain

$$\mathbf{S_w}^{-1}\mathbf{S_b}\mathbf{W}^* = \lambda \mathbf{W}^*. \tag{2.7}$$

Therefore, the sought-after projection matrix $\mathbf{W}^*$ is composed of the eigenvectors corresponding to the largest eigenvalues of $\mathbf{S_w}^{-1}\mathbf{S_b}$. To achieve satisfactory performance of LDA in different applications, researchers have proposed many extended LDA algorithms. For these LDA extensions, there are three main categories. The first category tries to enhance the performance of the standard LDA on nonlinear data. As the standard LDA is a linear optimisation method, it has relatively poor performance on the nonlinear data. Therefore, the methods, like nonparametric discriminant analysis [73], Kernel LDA [86] and subclass discriminant analysis (SDA) [137] are proposed. Nonparametric discriminant analysis obtains the expression ability of nonlinear data by the nonparametric extensions of the between and within scatter matrices. Kernel LDA solves this problem by kernel functions and maps the linearly inseparable data to a high-dimensional space where data become linearly separable. To well separate nonlinear data, SDA divides each class into a set of subclasses and tries to separate them at subclass-level instead of class-level. Compared with the standard LDA, the methods of this category shows better performance on nonlinear data. The second category focuses on solving the small sample size (SSS) problem, which occurs when the dimension of samples is greater than the number of samples. The SSS problem makes the within-class scatter matrix S_w singular so that the sought-after projection matrix W^* can't be obtained by using $S_w^{-1}S_b$. The representative

LDA extensions in this category include PCA+LDA [7], Regularised LDA (RLDA) [35] and Null LDA (NLDA) [16]. PCA+LDA solves the problem by reducing the dimensionality with PCA first to make the data dimension not greater than the number of samples and then applies LDA. RLDA solves the SSS problem of LDA by adding a small perturbation to S_w and makes S_w non-singular. Similar to PCA+LDA, NLDA also overcomes the SSS problem in two stages. NLDA first projects the original data into the null space of S_w, then obtains W^* by maximising $|W^T S_b W|$, where $|\cdot|$ denotes the determinant. The last category consists of the incremental versions of LDA which are aimed at online learning tasks. The methods in this category include QR decomposition-based incremental LDA [130], sequential ILDA and chunk ILDA [95], and incremental subclass discriminant analysis (ISDA) [17]. These methods focus on how to update the LDA matrix in a fast way when new data are added to the data set. In this case, the standard LDA needs to recalculate the LDA matrix based on all data while these methods only require very little computation.

Compared with the unsupervised learning of PCA and ICA, LDA extracts features in a supervised way, which utilises the classificatory information to construct the objective function. It is generally believed that when solving the problem of supervised classification, the performance of LDA tends to be better than that of PCA and ICA [48, 112, 12], because the former optimises the low-dimensional representation of the original data by focusing on maximising the separation of different classes, whilst PCA focuses on maximally keeping the information of the original data, and ICA is dedicated to minimising dependencies of the features in the original data. However, due to the limitation/assumption that every class has a normal distribution and shares the same covariance matrix, the classification performance of LDA suffers degradation when it deals with nonlinear data [137, 73, 119]. In this thesis, one of our goals is to develop cluster-based feature extraction methods based on LDA to overcome this limitation and obtain high classification accuracy.

2.3 Classification Learning

In the process of classification learning, a classifier is used to learn the classification function/decision boundary to map the inputs to a specific class. In supervised classification, a classifier usually learns the classification function by modelling the labelled inputs, where these labelled inputs are often called training data. There have been a number of classifiers developed, which can be roughly divided into two categories: generative classifiers and discriminative classifiers [92]. Generative classifiers refer to kinds of classifiers that model the joint probability of the input $\mathbf{x}$ and the label c, and make their predictions by employing Bayes rules to calculate the posterior probability. Then, generative classifiers assign a new input into the class with the highest posterior probability. The typical generative classifiers include naive Bayes (NB) [133], hidden Markov models (HMM) [100], and Bayesian networks (BN) [54]. In contrast, discriminative classifiers are obtained by modelling the posterior probability directly or by learning a direct mapping from data to class labels. Linear discriminant analysis classifier (LDAC) [33, 85], support vector machine (SVM) [111], *k*-nearest neighbour (*k*NN) [21], decision tree (DT) [78], and multilayer perceptron (MLP) [110] are well-known discriminative classifiers. In this section, NB, LDAC, SVM and MLP are introduced, which are commonly used and closely related to our work in Chapter V.

2.3.1 Naive Bayes

Naive Bayes (NB) classifiers are supervised classification algorithms based on Bayes rules and the assumption of conditional independence between each pair of features/attributes. Given a class label c and a sample $\mathbf{x}$ with d features, the probability of $\mathbf{x}$ belonging to class c is the posterior probability, and denoted as $P(y = c|\mathbf{x})$. According to Bayes rules, the posterior probability is calculated as follows:

$$P(y = c|\mathbf{x}) = \frac{P(\mathbf{x}|y = c)P(c)}{P(\mathbf{x})}, \tag{2.8}$$

where $c \in \{1, 2, 3, \ldots, C\}$ denotes the class label, C is the number of classes, $P(\mathbf{x}|y = c)$ is the likelihood, $P(c) = \frac{N_c}{N}$ is the priori probability that reflects the prior knowledge about class c, N_c is the number of samples in class c, and N is the number of samples in a training data set. $P(\mathbf{x}) = \sum_{c=1}^{C} P(\mathbf{x}|y = c)P(c)$ [29] is used to scale the expressions in Equation (2.8), and it is common for all samples. Thus, in practice, $P(y = c|\mathbf{x})$ is often calculated without $P(\mathbf{x})$ [117], which results in

$$\begin{aligned} P(y = c|\mathbf{x}) &= \frac{P(\mathbf{x}|y = c)P(c)}{P(\mathbf{x})} \\ &\propto P(\mathbf{x}|y = c)P(c). \end{aligned} \tag{2.9}$$

Furthermore, naive Bayes classifiers utilise the assumption that every feature is conditionally independent, resulting in the calculation of the likelihood $P(\mathbf{x}|y = c)$ being simplified to

$$P(\mathbf{x}|y = c) = \prod_{i=1}^{d} P(\mathbf{x}^i|y = c), \tag{2.10}$$

where $\mathbf{x}^i$ denotes the ith feature of sample $\mathbf{x}$. Combining Equations (2.9) and (2.10), the posterior probability is calculated as

$$P(y = c|\mathbf{x}) = P(c) \prod_{i=1}^{d} P(\mathbf{x}^i|y = c). \tag{2.11}$$

Finally, the class label of $\mathbf{x}$ is the label c that maximises the posterior probability. Thus, the classification function $f_{NB}(\mathbf{x})$ learnt by naive Bayes classifiers is obtained and formulated as follows:

$$f_{NB}(\mathbf{x}) = \underset{c}{\operatorname{argmax}}\, P(c) \prod_{i=1}^{d} P(\mathbf{x}^i|y = c). \tag{2.12}$$

Typically, the maximum a posteriori (MAP) method is used to estimate $P(\mathbf{x}^i|y = c)$. There have been many different naïve Bayes classifiers produced due to different assumptions being applied to the distribution of $P(\mathbf{x}^i|y = c)$. Representative naïve Bayes

classifiers include: Gaussian NB, assumes that the distribution of every feature is Gaussian, and is used for classifying Gaussian distributed data; Multinomial NB, assumes that features of data follow multinomially distribution, and it is often used to separate classes with discrete features; Bernoulli NB, which was proposed for dealing with binary-valued features; and kernel NB, which tries to learn the distribution of features by kernel density estimation.

2.3.2 Linear Discriminant Analysis Classifier

Linear discriminant analysis classifier (LDAC) is a probabilistic discriminative classifier. Unlike NB classifiers, which indirectly obtain the posterior probability, LDAC directly calculates the posterior probability $P(y = c|\mathbf{x})$ under the assumption that each class has a normal distribution and has the same covariance matrix, namely $P(\mathbf{x}|y = c) \sim \mathcal{N}(\mu_c, \Sigma)$. Thus, the likelihood $P(\mathbf{x}|y = c)$ can be calculated as

$$P(\mathbf{x}|y = c) = \frac{1}{\sqrt{(2\pi)^d|\Sigma|}} exp\big(-\frac{1}{2}(\mathbf{x} - \mu_c)^T\Sigma^{-1}(\mathbf{x} - \mu_c)\big), \tag{2.13}$$

where μ_c denotes the mean of class c, Σ denotes the common covariance matrix, d is the number of features of sample $\mathbf{x}$, and $|\Sigma|$ and Σ^{-1} are the determinant and inverse of the common covariance matrix, respectively. Therefore, based on Equations (2.9) and (2.13), the posterior probability $P(y = c|\mathbf{x})$ can be directly calculated in LDAC as:

$$P(y = c|\mathbf{x}) = P(c)\frac{1}{\sqrt{(2\pi)^d|\Sigma_c|}} exp\big(-\frac{1}{2}(\mathbf{x} - \mu_c)^T\Sigma_c^{-1}(\mathbf{x} - \mu_c)\big). \tag{2.14}$$

Finally, LDAC obtains a classification function $f_{LDAC}(\mathbf{x})$ between class c and class c', where $c \neq c'$, c and $c' \in \{1, 2, 3, \ldots, C\}$ as the following:

$$
\begin{aligned}
f_{LDAC}(\mathbf{x}) &= ln(P(y=c|\mathbf{x})) - ln(P(y=c'|\mathbf{x}) \\
&= (\mu_c^T - \mu_c'^T)\Sigma^{-1}\mathbf{x} - \frac{1}{2}(\mu_c^T\Sigma^{-1}\mu_c) \\
&\quad - (\mu_c'^T\Sigma^{-1}\mu_c') + ln\frac{P(c)}{P(c')},
\end{aligned} \tag{2.15}
$$

if we denote

$$
\mathbf{w^T} = (\mu_c^T - \mu_c'^T)\Sigma^{-1}, \quad \text{and}
$$

$$
b = -\frac{1}{2}(\mu_c^T\Sigma^{-1}\mu_c) - (\mu_c'^T\Sigma^{-1}\mu_c') + ln\frac{P(c)}{P(c')},
$$

then we obtain $f_{LDAC}(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$. So, the classification function of LDAC is linear, where $\mathbf{w}$ and b can be interpreted as the normal vector and intercept of $f_{LDAC}(\mathbf{x})$, respectively. Additionally, similarly to NB classifiers, LDAC assigns sample $\mathbf{x}$ with the class label that corresponds to the largest posterior probability.

2.3.3 Support Vector Machine

In contrast to LDAC, support vector machine (SVM) is a non-probabilistic discriminative classifier, which obtains the classification function through directly modelling the mapping from input $\mathbf{x}$ to the class label c. The goal of SVM is to find an optimal separating hyperplane $\mathbf{w}^T\mathbf{x} + b = 0$ that can maximise the margin between two classes. This can be achieved by optimising the following formulation

$$
\begin{aligned}
&\underset{\mathbf{w}}{\operatorname{argmin}} \frac{\|\mathbf{w}\|^2}{2} \\
s.t. \qquad & 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b) \leq 0 \qquad \forall i,
\end{aligned} \tag{2.16}
$$

where $\mathbf{x}_i$ is the ith sample, $c_i \in \{1, -1\}$ is the class label of $\mathbf{x}_i$, $\mathbf{w}$ is the normal vector of the hyperlane, and $\|\mathbf{w}\|^2$ denotes the inverse distance between the marginal hyperplanes $\mathbf{w}^T\mathbf{x} + b = 1$ and $\mathbf{w}^T\mathbf{x} + b = -1$.

To solve this optimisation problem, SVM employs the Karush-Kuhn-Tucker (KKT) [64] approach, which results in

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i c_i \mathbf{x}_i$$

$$s.t. \qquad \alpha_i \geq 0 \quad \text{and} \quad \sum_{i=1}^{N} \alpha_i c_i = 0 \quad \forall i. \tag{2.17}$$

Therefore, the classification function learnt by SVM is defined by

$$f_{SVM}(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b = \sum_{i=1}^{N} \alpha_i c_i \langle \mathbf{x}_i, \mathbf{x} \rangle + b \tag{2.18}$$

$$s.t. \qquad \alpha_i \geq 0 \quad \text{and} \quad \sum_{i=1}^{N} \alpha_i c_i = 0 \quad \forall i,$$

where α_i is a Lagrangian multiplier. According to Equation (2.18), it is readily noted that the classification function of SVM is also linear. To effectively solve for more complex data, the linear SVM is often extended to nonlinear SVM (non-SVM). Nonlinear SVM can be obtained easily using a kernel function $K(\mathbf{x_i}, \mathbf{x_j})$. The main idea of the kernel functions is to implicitly map the original data into a higher feature space $\mathcal{F}$ and calculate the inner product between two mapped features $\Phi(\mathbf{x}_i)$ and $\Phi(\mathbf{x}_j)$ in $\mathcal{F}$ as $K(\mathbf{x_i}, \mathbf{x_j}) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle$. Thus, the resulting classification function of non-SVM is represented by

$$f_{non-SVM}(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b = \sum_{i=1}^{N} \alpha_i c_i K(\mathbf{x}_i, \mathbf{x}) + b \tag{2.19}$$

$$s.t. \qquad \alpha_i \geq 0 \quad \text{and} \quad \sum_{i=1}^{N} \alpha_i c_i = 0 \quad \forall i.$$

In non-SVM, the commonly used kernel functions include the radial basis kernel function (RBF) and the polynomial kernel function (Poly) [10]. In addition, it is known that SVM and non-SVM were originally designed for binary classification problems. To handle the multiclass classification problem, the *one-against-one* scheme and

one-against-all scheme [61, 62] are utilised.

2.3.4 Multilayer Perceptron

Multilayer perceptron (MLP) is another non-probabilistic discriminative classifier. Similar to SVM, it obtains the classification function through directly modelling the mapping from input $\mathbf{x}$ to the class label c. An MLP is a kind of feedforward artificial neural network. It consists of an input layer to receive the sample, an output layer for making a decision or prediction about the input, and an arbitrary number of hidden layers between the input layer and the output layer. In practice, the three-layered MLP (an input layer, one hidden layer and an output layer) is commonly used, since a three-layered MLP can approximate any mapping relations between inputs and outputs [102, 115].

The classification function ($f_{MLP}(\mathbf{x})$) [18] of a three-layered MLP can be written as

$$f_{MLP}(\mathbf{x}) = \mathbf{w}_{out}^T h(\mathbf{x}) + b_{out}, \tag{2.20}$$

where $\mathbf{w}_{out}$ denotes the weight vector of the output layer, b_{out} is the bias of the output layer, and $h()$ is the transfer/activation function in the hidden layer. Based on different choices of $h()$, different MLP classifiers are generated. The widely used transfer functions include the sigmoid function, hyperbolic tangent sigmoid function, and rectified linear unit (ReLU) function. To learn the optimal classification function, MLP employs the *gradient descent* method to minimise a given criterion $Q(f_{MLP}(\mathbf{x}))$, which is usually the *Mean Squared Error* or *Cross-Entropy* criterion [8].

2.4 Summary

In this chapter, we have reviewed many feature extraction methods and classifiers that are widely used in supervised classification. In terms of approaches for extracting features, deep feature extraction methods and traditional feature extraction methods are

introduced. We briefly introduce some popular deep feature extraction methods in the field of face recognition, text classification, human activity recognition and data generation. Then, we emphasise the traditional feature extraction methods and provide a detailed introduction to three classical ones: PCA, ICA and LDA. Compared with deep feature extraction methods, traditional methods do not require high computation time or massive storage. Moreover, they are supported by forceful theory and strong interpretability. After this, a review of classifiers is provided. As an algorithm for obtaining class labels for samples, the quality of the classifier undoubtedly affects the classification performance of supervised classification. In Section 2.3, some typical generative and discriminative classifiers are presented, including NB classifiers, LDAC, SVM and MLP. Based on the analysis and description in this chapter, we note that the feature extraction method and the classifier are two important factors in achieving high classification accuracy. Thus, in the rest of this thesis, our contributions to feature extraction and classifiers are presented, which improve classification performance by using the idea of clusters/subclasses.

CHAPTER III

Within-class Multimodal Classification

This thesis aims to develop cluster-based feature extraction methods and classifiers to improve the classification performance of supervised classification. This chapter focuses on cluster-based feature extraction. A comprehensive study on within-class multimodal data distribution is presented, which is guided by five key questions about within-class multimodality. Through systematic experimentation using various artificial and real-world data, the questions are answered and some interesting findings are obtained.

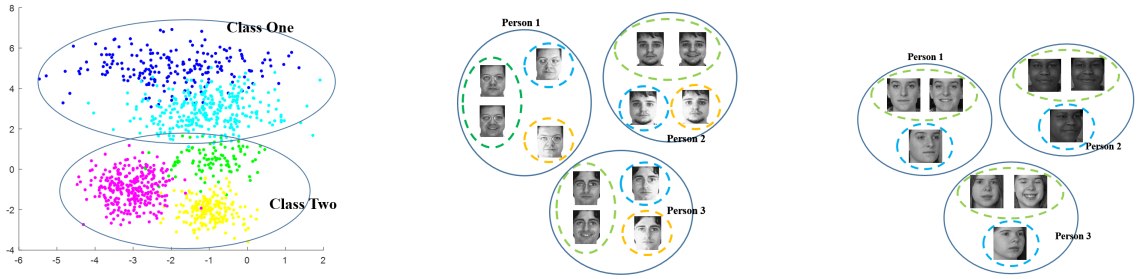
3.1 Motivation

Understanding the underlying data distribution before applying a machine learning process is an important step in the analysis of data, as otherwise, wrong choices may be made in the different stages of the machine learning process. Every single algorithm used in machine learning has, either explicitly or implicitly, some assumptions about the data for it to work effectively. For linear regression, the typical assumptions include linearity (there is a linear relationship between the independent and dependent variables), exogeneity (the errors between observed and predicted values should have conditional mean zero), multicollinearity (the independent variables must all be linearly independent), homoscedasticity (the errors have the same variance in each observation), and normality (the errors have a normal distribution) [45, 107]. For random forests [13], one assumption

is that changes in the dependent variable are best described by hyper-rectangles in the independent variables (because they are based on trees). Another assumption is that no future value of the dependent variable will be outside the range of values already in the training data. If the distribution of data can be described as the canonical statistical distributions, it is possible to gain much inferential and predictive power [79]. The key to any successful use of data in an analysis or in making a decision is applying the correct machine learning/statistical modelling technique to the data at hand.

In this chapter we consider a particular type of data distribution where there are multiple modalities (concentrations/clusters of data) within each class, *within-class multimodality*, and study how to choose the most appropriate feature extraction methods to model such data more effectively. Figure 3.1(a) illustrates within-class multimodality at a conceptual level, where there are two and three modalities, respectively, in Class One and Class Two. Within-class multimodality is prevalent in the real world. For example, we can recognise people under different illuminations, and also in different poses. If we represent face images of the same person under different illuminations, it is likely that different images with different illuminations will be in different clusters (see Figure 3.1(b) for an illustration). Actually, face recognition under varying illuminations is a challenging problem[135, 118]. The same can be said of face recognition from different head poses (see Figure 3.1(c) for an illustration). Another potential application is energy disaggregation of appliances by non-intrusive load monitoring (NILM) [90], namely disaggregating the total consumption readings into the consumption patterns of each individual appliance, where the total consumption reading of a house represents a class and the appliances in a house are the modalities within this class. Therefore, dividing a class into multiple modalities is similar to disaggregating the total consumption of all appliances into the consumption of each appliance.

Within-class multimodality has been largely ignored in the literature, or at least under-studied. The closest studies are *linear discriminant analysis* (LDA) [33, 101],



(a) Modalities in two classes

(b) Examples of illumination modalities existing in face images

(c) Examples of head pose modalities existing in face images

Figure 3.1: Illustration of within-class multimodality. (a) There are two modalities in Class One, and three modalities in Class Two, where different modalities are denoted by different colours. (b) Each person has three different illumination modalities: two face images in the green dotted circle are taken under normal lighting; one face image in the cyan dotted circle is taken under normal lighting and right light on; one face image in the orange dotted circle is taken under normal lighting and left light on. (c) Each person has two different head pose modalities: two face images in the green dotted circle are taken with frontal head pose, and one face image in the cyan dotted circle is taken with rightwards head pose.

subclass discriminant analysis (SDA) [137] and *separability-oriented subclass discriminant analysis* (SSDA) [119]. LDA is a classical approach to discriminant dimensionality reduction. It transforms data from the original data space into a lower dimensional space (LDA space) so that the within-class compactness is maximised whilst the between-class separation is maximised. This is achieved through maximising the well-known Fisher objective, which is composed by the within-class scatter matrix and between-class scatter matrix [33, 101]. In the presence of within-class multimodality, LDA reduces dimensionality by merging multiple modalities in each class into a single modality. SDA extends LDA in order to separate classes at a subclass level rather than at a class level. It transforms data into a lower dimensional SDA space so that the between-subclass separation is maximised, and within-class compactness is maximised. The SDA subclasses are discovered using the *leave-one-out-test* (LOOT) criterion proposed in [137] or the *stability criterion* [83]. SSDA extends SDA to minimise the level of overlap between subclasses within every class; thus the between-class separation is

maximised, between-subclass separation is maximised, and within-class compactness is maximised. The SSDA subclasses are discovered by the agglomerative hierarchical clustering algorithm using a new criterion called the separability criterion [119], which aims to divide each class into several non-overlapping clusters.

Within-class unimodality classification[1] is well understood, where the aim is to build a model assuming there is one modality per class. It is well-known that simultaneously minimising intra-class variance and maximising inter-class variance will increase learning performance [31, 122, 123]. However, not enough is known about *within-class multimodality classification*, when the data distribution is within-class multimodal. In this chapter, we present a study on within-class multimodality classification as guided by the following questions:

- Question 1: Is it necessary to address within-class multimodality?
- Question 2: How many within-class modalities should we use?
- Question 3: How should we utilise the modalities?
- Question 4: Does considering within-class multimodality bring real benefit?
- Question 5: If we keep increasing the number of modalities, what will happen?

The study of these questions is important for a number of reasons. Firstly, it will reveal a relationship between the modality of the data distribution and the comparative performance of the classification, so it is possible to gain an insight into the data through the comparative model performance using different data dimensionality reduction techniques. Secondly, it will establish the fact that different dimensionality reduction techniques are suitable for different data distributions. Thirdly, it will provide a direction for improving other machine learning algorithms such as neural networks by designing new loss functions.

[1] Unimodality is when data distribution has one centre of concentration, whereas multimodality is when data distribution has multiple centres of concentration.

We create artificial data sets having a range of modalities and conduct extensive experiments in order to answer Questions 1-3 (and possibly Question 5). We also select real world data sets that clearly have multiple modalities and conduct extensive experiments to answer Question 4.

The rest of the chapter is organised as follows. Section 3.2 presents relevant work considering within-class modalities, including subclass discriminant analysis (SDA) and separability-oriented subclass discriminant analysis (SSDA). Section 3.3 attempts to answer various questions about multimodality using artificial data sets, and Section 3.4 attempts to answer other questions using real data sets. Section 3.5 concludes the chapter with a summary.

Additionally, in the rest of this chapter we use cluster, subclass and modality in different contexts but these terms are interchangeable.

3.2 Related Work

In this section, we present an overview of related work, including the SDA and SSDA, to provide the context for this study and introduce the necessary technical notations.

3.2.1 Subclass Discriminant Analysis

Subclass discriminant analysis (SDA) [137] is a variant of LDA that separates classes at a subclass level rather than at a class level, based on the observation that the data distribution in a class may be multimodal (i.e., forming clusters). This is achieved by dividing each class into a set of subclasses and then running an LDA-like optimisation process to obtain a projection matrix $\mathbf{W}$ by maximising between-subclass separation and within-class compactness.

The between-class scatter matrix $\mathbf{S_b}$ of LDA is replaced by the between-subclass scatter

matrix, which is defined below in Equation (3.1):

$$\mathbf{S_b^{SDA}} = \sum_{i=1}^{C-1} \sum_{j=1}^{K_i} \sum_{l=i+1}^{C} \sum_{n=1}^{K_l} p_{ij} p_{ln} (\mu_{ij} - \mu_{ln})(\mu_{ij} - \mu_{ln})^T, \tag{3.1}$$

where C denotes the number of classes, K_i (K_l) denotes the number of subclasses in class i (l), μ_{ij} (μ_{ln}) denotes the mean of the jth (nth) subclass in class i (l), $p_{ij} = \frac{N_{ij}}{N}$ ($p_{ln} = \frac{N_{ln}}{N}$) denotes the prior of the jth (n)th subclass of class i (l), N is the number of training samples and N_{ij} (N_{ln}) is the number of samples in the jth (nth) subclass of class i (l).

The within-class scatter matrix of SDA is defined as the sample covariance matrix as below in Equation (3.2):

$$\mathbf{S_w^{SDA}} = \frac{1}{N} \sum_{j=1}^{N} (\mathbf{x}_j - \mu)(\mathbf{x}_j - \mu)^T, \tag{3.2}$$

where N, $\mathbf{x}_j$, and μ are the number of training samples, the jth training sample, and the mean of all training samples, respectively.

The Fisher objective of SDA is defined as follows (Equation (3.3)):

$$J^{SDA}(\mathbf{W}) = \frac{tr(\mathbf{W}^T \mathbf{S_b^{SDA}} \mathbf{W})}{tr(\mathbf{W}^T \mathbf{S_w^{SDA}} \mathbf{W})}. \tag{3.3}$$

In order to divide each class into the same number of subclasses, a *leave-one-out-test* (LOOT) criterion [137] or a faster *stability criterion* [83] is used together with a nearest neighbour based clustering algorithm [137]. Firstly, the clustering algorithm is used to sort the samples of each class so that samples with smaller Euclidean distance stay closer. To achieve this, two samples A and B are found in each class that have the largest Euclidean distance between each other, and are taken as the 1st and nth samples in the sorted data, respectively. After that, the samples ranked from 1st to $(n/2)$th are near A, and the samples ranked from $(n/2 + 1)$th to nth are near B. Then, based on the number of subclasses set by the user, the sorted samples are divided into the specified number of subclasses for each

class. Finally, the LOOT criterion or stability criterion is used to find the optimal number of subclasses for each class.

3.2.2 Separability-oriented Subclass Discriminant Analysis

Separability-oriented subclass discriminant analysis (SSDA) [119] is an extension of SDA, which also separates classes at subclass level. It aims to (1) maximise the between-subclass separation within every class; (2) maximise the within-class compactness; and (3) maximise the overall between-class separation. This is achieved through an LDA-like optimisation process operating at subclass level and with a different Fisher objective.

The way to find optimal subclasses for each class is very different from SDA. SSDA aims to find subclasses with no or little overlap through agglomerative hierarchical clustering guided by a *separability criterion* [119]. The resulting clustering is one that maximises the average Euclidean distance (*AED*) between the mean of a class and the means of subclasses in the class.

Three versions of SSDA exist [119], each having a different combination of between-class scatter matrix and within-class scatter matrix. One version is reviewed here. The between-class scatter matrix in SSDA, $\mathbf{S_b^{SSDA}}$, is defined in terms of the subclasses:

$$\mathbf{S_b^{SSDA}} = \sum_{i=1}^{C} \frac{N_i}{N} \sum_{j=1}^{K_i} (\mu_{ij} - \mu)(\mu_{ij} - \mu)^T, \tag{3.4}$$

where N is the number of training samples, N_i is the number of samples in class i ($i = 1, 2, \ldots, C$, C is the number of classes) such that $\sum_{i=1}^{C} N_i = N$, K_i is the number of subclasses in class i, μ is the mean of all training samples, and μ_{ij} is the mean of subclass j of class i.

The within-class scatter matrix is the standard LDA within-class matrix, $\mathbf{S_w^{SSDA}} = \mathbf{S_w}$. Therefore, the Fisher objective of SSDA, $J^{SSDA}(\mathbf{W})$ is below, replacing $\mathbf{S_b}$ by $\mathbf{S_b^{SSDA}}$. Moreover, we summarise the idea of SSDA in Algorithm 1 and show the main steps of the

SSDA algorithm using a flowchart in Figure 3.2. Here, the notations used in the flowchart have the same meaning as those used in Algorithm 1.

$$J^{SSDA}(\mathbf{W}) = \frac{tr(\mathbf{W}^T\mathbf{S}_\mathbf{b}^{\mathbf{SSDA}}\mathbf{W})}{tr(\mathbf{W}^T\mathbf{S}_\mathbf{w}^{\mathbf{SSDA}}\mathbf{W})} = \frac{tr(\mathbf{W}^T\mathbf{S}_\mathbf{b}^{\mathbf{SSDA}}\mathbf{W})}{tr(\mathbf{W}^T\mathbf{S}_\mathbf{w}\mathbf{W})}. \tag{3.5}$$

Algorithm 1 SSDA: In this algorithm, C is the number of classes, AED_{ik} is the average Euclidean distance between the mean of class i and the means of subclasses in class i, and K_i^* is the number of subclasses found by SSDA for class i

Input: A set of training samples $\mathbf{X}$ with class labels and the maximum number of subclasses K.

Output: Projection matrix $\mathbf{W}^*$.

1: **for** i = 1 to C **do**
2: **for** k = 1 to K **do**
3: Calculate AED_{ik} using the agglomerative hierarchical clustering guided by a *separability criterion*.
4: **end for**
5: K_i^* = $argmax_k(AED_{ik})$.
6: Calculate $\mathbf{S}_\mathbf{b}^{\mathbf{SSDA}}$ with K_i^* subclasses using Eq.(3.4).
7: Calculate $\mathbf{S}_\mathbf{w}$ using Eq.(2.5).
8: **end for**
9: The columns of transformation matrix $\mathbf{W}^*$ is given by the eigenvectors corresponding to the largest eigenvalues of $\mathbf{S}_\mathbf{w}^{-\mathbf{1}}\mathbf{S}_\mathbf{b}^{\mathbf{SSDA}}$.

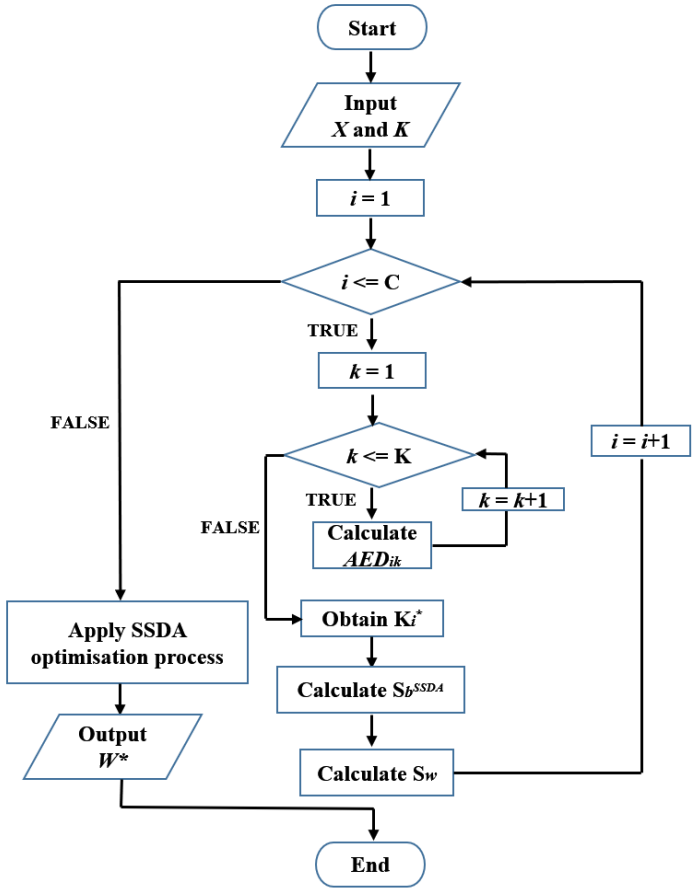


Figure 3.2: The flowchart of SSDA algorithm.

3.3 Multimodality in Artificial Data

3.3.1 Details of Artificial Data

In order to answer the research questions mentioned above, we generate four types of artificial data.

- Type 1 consists of two different classes, and samples in each class are from a single multivariate normal distribution. This type is denoted by C2M1.
- Type 2 consists of two different classes, and every class has two subclasses of samples generated from two multivariate normal distributions. This type is denoted by C2M2.

- Type 3 consists of two different classes, and every class has three subclasses of samples generated from three multivariate normal distributions. This type is denoted by C2M3.

- Type 4 consists of three different classes, and every class has three subclasses of samples generated from three multivariate normal distributions. This type is denoted by C3M3.

The number of variables is one parameter in a multivariate normal distribution, which is empirically set to 30 for all types of artificial data in our studies. Two other important parameters are: the mean μ and covariance $\mathbf{\Sigma}$, which are needed to generate artificial data from a multivariate normal distribution. In our studies, the mean μ is a 1-by-30 vector and the values of the mean vector are integers chosen randomly from the range $[1, 10]$. Covariance $\mathbf{\Sigma}$ is a 30-by-30 diagonal matrix. There are two covariance matrices for C2M1, one for each class. The values of one covariance matrix for C2M1 are integers chosen randomly from the range $[10, 21]$, and the values of the other covariance matrix are integers chosen randomly from the range $[20, 41]$.

There are four covariance matrices for C2M2, one for each subclass and two for each class (there are two subclasses in each class). For class one, the values of the covariance matrices for the two subclasses are integers chosen randomly from the range $[10, 21]$, and the values of the covariance matrices for the two subclasses of class two are integers chosen randomly from the range $[20, 41]$.

There are six covariance matrices for C2M3, one for each subclass and three for each class. For class one, the values of the covariance matrices for the three subclasses are integers chosen from the range $[10, 21]$ randomly. For class two, the values of the covariance matrices for the three subclasses are integers chosen randomly from the range $[20, 41]$.

There are nine covariance matrices for C3M3, one for each subclass and three for each class. For class one, the values of the covariance matrices for the three subclasses are

integers chosen from the range $[1, 10]$ randomly. For class two and class three, the values of the covariance matrices for the three subclasses are integers chosen randomly from the ranges $[10, 21]$ and $[20, 41]$, respectively.

In total, 10 data sets are generated for each type, and every class of every artificial data set (any type) has 1000 samples. Therefore, C2M1, C2M2 and C2M3 each has a total of 2000 samples with 1000 per class. For C2M2 and C2M3, the samples in each class are randomly placed into two and three subclasses, respectively, according to a probability distribution that varies from data set 1 to 10. C3M3 has a total of 3000 samples with 1000 per class. The samples in each class are randomly placed into three subclasses in the same way as for C2M2 and C2M3. The actual numbers of samples per subclass are shown in Table 3.2, Table 3.3 and Table 3.4.

Multiple modalities exist in data. In order to have full insight about the issue of within-class multimodality, various questions can be asked and answered. In Section 3.1, some questions are posed explicitly, and the rest of this chapter is to seek answers to these questions. Some questions will be answered using artificial data in this section. Other questions will be answered using real-world data in the next section.

3.3.2 Q1: Is it necessary to address within-class multimodality?

To answer this question we consider and compare experimentally three approaches in the presence of within-class multimodality:

- separating within-class modalities for every class through the extraction of features by dimensionality reduction methods such as SDA and SSDA;
- merging within-class modalities as a uni-modality for every class in the process of feature extraction using a dimensionality reduction method such as LDA; and
- doing nothing about within-class multimodality and using the original data for classification.

In order to evaluate these three approaches, we conduct experiments using k-nearest neighbour (kNN, k=1) as the classifier on all of the artificial data sets. We consider four cases: (1) Original: the original artificial data sets without any processing for dimensionality reduction; (2) LDA processed; (3) SDA processed; and (4) SSDA processed. In addition, we use one half of each data set for training and the other half for testing.

Table 3.1, Table 3.2, Table 3.3 and Table 3.4 show the experimental results in the four cases on all of the artificial data sets. From these results, we can observe the following:

- It is apparent that SSDA outperforms Original and LDA on all artificial data sets. In particular, SSDA improves the classification accuracy over Original by at least 9% on all of the C2M1, C2M2 and C2M3 data sets, and by at least 14% on the C3M3 data sets.

- LDA, SDA and SSDA outperform Original consistently, so dimensionality reduction in the style of LDA can indeed improve classification performance significantly. Whilst this finding is not new, it indicates that doing nothing about multimodality is suboptimal.

- When there is only one modality per class: From Table 3.1, we can see that both SDA and SSDA obtain the same classification accuracy as LDA on the data sets C2M1-5 and C2M1-10, which is caused by both SDA and SSDA only find one subclass for each class of the two data sets. In addition, the differences between LDA, SDA and SSDA on the rest of C2M1 data sets do not appear to be significant. This suggests that when there is only one modality per class, doing dimensionality reduction using SDA or SSDA makes little difference from using LDA.

- As for LDA and its variants, we can rank order them in terms of their performance: LDA$\leq$SDA$\leq$SSDA on the artificial data sets with within-class multimodality,

namely C2M2, C2M3 and C3M3. This suggests that dealing with within-class multimodality using SSDA is the best approach.

- When there are multiple modalities per class: from Table 3.2, Table 3.3 and Table 3.4, it is clear that doing dimensionality reduction at the subclass level as in SDA or SSDA is better than at the class level as in LDA. Furthermore, SSDA clearly outperforms SDA in these experiments. This suggests that separating subclasses (in other words, reducing the overlap of different subclasses) within every class and at the same time separating all classes is a better approach than simply separating subclasses in a class from the subclasses of other classes.

- When the number of modalities per class increases: according to Table 3.1, Table 3.2 and Table 3.3, in general the classification accuracy drops in all methods, suggesting that the complexity of the problem increases. This can be seen more clearly in Figure 3.3. Interestingly, the margin of performance drop is the smallest with SSDA, suggesting that SSDA is more robust than Original, LDA and SDA when the number of modalities per class changes.

From these observations we can draw the conclusion that it is indeed necessary to deal with the issue of within-class multimodality. Furthermore, this conclusion will be confirmed by using the real data sets in Section 3.4.

Table 3.1: Classification accuracy with kNN (k=1) of Original, LDA, SDA and SSDA on ten C2M1 data sets

Methods / Data sets	Original	LDA	SDA	SSDA
C2M1-1	0.8700	0.9700	**0.9750**	0.9700
C2M1-2	0.8590	0.9540	**0.9640**	0.9540
C2M1-3	0.8430	0.9500	0.9580	**0.9660**
C2M1-4	0.8180	0.9490	**0.9610**	0.9540
C2M1-5	0.8540	**0.9540**	**0.9540**	**0.9540**
C2M1-6	0.8730	0.9620	0.9650	**0.9660**
C2M1-7	0.8730	0.9670	**0.9750**	0.9690
C2M1-8	0.8630	0.9660	0.9660	**0.9700**
C2M1-9	0.8170	0.9320	**0.9380**	0.9320
C2M1-10	0.8590	**0.9620**	**0.9620**	**0.9620**

Table 3.2: Classification accuracy with kNN (k=1) of Original, LDA, SDA and SSDA on ten C2M2 data sets, along with the ratio between the numbers of samples from different subclasses in each class.

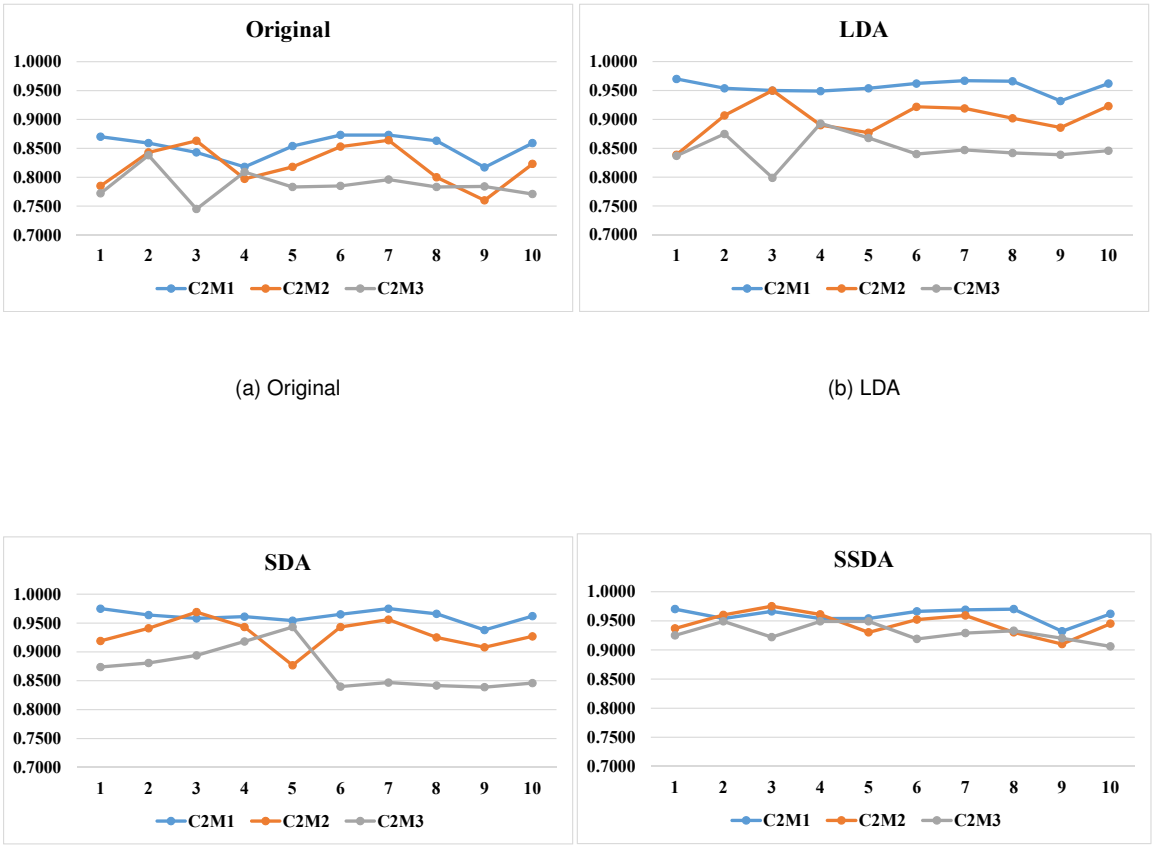
Methods / Data sets	Original	LDA	SDA	SSDA	ratio	
					Class One	Class Two
C2M2-1	0.7850	0.8390	0.9190	**0.9370**	684:316	701:299
C2M2-2	0.8430	0.9070	0.9410	**0.9600**	676:324	693:307
C2M2-3	0.8630	0.9500	0.9690	**0.9750**	521:479	508:492
C2M2-4	0.7970	0.8900	0.9430	**0.9610**	479:521	499:501
C2M2-5	0.8180	0.8770	0.8770	**0.9300**	491:509	497:503
C2M2-6	0.8530	0.9220	0.9430	**0.9520**	486:514	512:488
C2M2-7	0.8640	0.9190	0.9560	**0.9590**	289:711	305:695
C2M2-8	0.8000	0.9020	0.9250	**0.9300**	274:726	294:706
C2M2-9	0.7600	0.8860	0.9080	**0.9100**	186:814	208:792
C2M2-10	0.8230	0.9230	0.9270	**0.9450**	793:207	796:204

Table 3.3: Classification accuracy with kNN (k=1) of Original, LDA, SDA and SSDA on ten C2M3 data sets, along with the ratio between the numbers of samples from different subclasses in each class.

Methods / Data sets	Original	LDA	SDA	SSDA	ratio	
					Class One	Class Two
C2M3-1	0.7720	0.8370	0.8740	**0.9250**	208:531:261	189:535:276
C2M3-2	0.8380	0.8750	0.8810	**0.9490**	359:187:454	362:216:422
C2M3-3	0.7450	0.7990	0.8940	**0.9220**	358:360:282	327:380:293
C2M3-4	0.8090	0.8930	0.9180	**0.9490**	141:354:505	130:398:472
C2M3-5	0.7830	0.8680	0.9430	**0.9490**	11:347:642	7:351:642
C2M3-6	0.7850	0.8400	0.8400	**0.9190**	8:347:645	2:343:655
C2M3-7	0.7960	0.8470	0.8470	**0.9290**	188:652:160	194:612:194
C2M3-8	0.7830	0.8420	0.8420	**0.9330**	437:394:169	438:403:159
C2M3-9	0.7840	0.8390	0.8390	**0.9200**	431:142:427	413:150:437
C2M3-10	0.7710	0.8460	0.8460	**0.9060**	426:161:413	452:147:401

Table 3.4: Classification accuracy with kNN (k=1) of Original, LDA, SDA and SSDA on ten C3M3 data sets, along with the ratio between the numbers of samples from different subclasses in each class.

Methods / Data sets	Original	LDA	SDA	SSDA	ratio		
					Class One	Class Two	Class Three
C3M3-1	0.7593	0.8480	0.8947	**0.9433**	659:127:214	654:155:191	666:131:203
C3M3-2	0.7740	0.8287	0.9053	**0.9413**	559:114:327	569:115:316	562:107:331
C3M3-3	0.7773	0.8587	0.9080	**0.9400**	776:147:77	785:152:63	736:174:90
C3M3-4	0.7120	0.7267	0.7267	**0.9220**	313:278:409	326:237:437	321:246:433
C3M3-5	0.7767	0.8067	0.8067	**0.9393**	330:262:408	265:287:448	317:284:399
C3M3-6	0.7273	0.7647	0.8800	**0.9253**	425:243:332	449:242:309	400:260:340
C3M3-7	0.7847	0.8180	0.8920	**0.9280**	168:435:397	195:425:380	176:441:383
C3M3-8	0.7720	0.8560	0.8560	**0.9413**	165:405:430	183:412:405	163:450:387
C3M3-9	0.7840	0.8680	0.8893	**0.9433**	36:613:351	46:622:332	47:601:352
C3M3-10	0.7987	0.8787	0.8787	**0.9520**	16:500:484	23:462:515	20:487:493



(a) Original

(b) LDA

(c) SDA

(d) SSDA

Figure 3.3: The classification performance of Original, LDA, SDA and SSDA on ten C2M1 data sets, ten C2M2 data sets and ten C2M3 data sets: In the line charts, the horizontal axis shows the ten data sets from C2M1, C2M2 and C2M3, and the vertical axis shows the classification accuracy.

3.3.3 Q2: How many within-class modalities should we use?

There is a clear difference between SDA and SSDA in terms of classification accuracy, as shown in Table 3.1, Table 3.2, Table 3.3 and Table 3.4. SDA and SSDA are both trying to separate classes at subclass level, but they are different in two ways: (1) how to find the within-class multimodalities; (2) once found, how to make use of these modalities. We examine the first issue in this subsection and discuss the second issue in Section 3.3.4 .

SDA uses a stability criterion to find class modalities, whereas SSDA uses a separability criterion. Table 3.5, Table 3.6, Table 3.7 and Table 3.8 show the numbers of class modalities found by SDA and SSDA for the 10 data sets, of type C2M1, C2M2, C2M3 and C3M3, respectively. It is clear that the numbers are quite different for SDA and SSDA. The numbers found by SSDA in general are quite close to the true numbers of within-class modalities, and identical in most of the data sets. Apart from in a few cases, the numbers found by SDA are quite different from the true numbers.

Furthermore, SSDA can even find true within-class modalities for classes with imbalanced proportions of data between subclasses. For example, SSDA separates each of Class One, Class Two and Class Three of C3M3-10 into three modalities, when their subclass ratios are $16 : 500 : 484$, $23 : 462 : 515$ and $20 : 487 : 493$ respectively.

All of these observations suggest that (1) steadily good classification performance is guaranteed by the correct number of modalities found; and (2) SSDA can find the number of within-class modalities more correctly than SDA, which will be verified on the two face databases in Section 3.4.2

Table 3.5: The number of subclasses found by SDA and SSDA for each class in the C2M1 data sets

Methods / Data sets	SDA		SSDA	
	Class One	Class Two	Class One	Class Two
C2M1-1	3	3	**1**	**1**
C2M1-2	3	3	**1**	**1**
C2M1-3	3	3	2	2
C2M1-4	4	4	2	2
C2M1-5	**1**	**1**	**1**	**1**
C2M1-6	3	3	2	2
C2M1-7	2	2	4	4
C2M1-8	**1**	**1**	2	2
C2M1-9	2	2	**1**	**1**
C2M1-10	**1**	**1**	**1**	**1**

Table 3.6: The number of subclasses found by SDA and SSDA for each class in the C2M2 data sets

Methods / Data sets	SDA		SSDA	
	Class One	Class Two	Class One	Class Two
C2M2-1	5	5	4	**2**
C2M2-2	6	6	**2**	**2**
C2M2-3	4	4	**2**	**2**
C2M2-4	6	6	**2**	**2**
C2M2-5	1	1	3	3
C2M2-6	4	4	**2**	**2**
C2M2-7	4	4	3	**2**
C2M2-8	6	6	**2**	**2**
C2M2-9	3	3	**2**	**2**
C2M2-10	**2**	**2**	4	**2**

Table 3.7: The number of subclasses found by SDA and SSDA for each class in the C2M3 data sets

Methods / Data sets	SDA		SSDA	
	Class One	Class Two	Class One	Class Two
C2M3-1	8	8	**3**	**3**
C2M3-2	10	10	**3**	**3**
C2M3-3	3	3	**3**	**3**
C2M3-4	6	6	2	**3**
C2M3-5	**3**	**3**	**3**	**3**
C2M3-6	15	15	**3**	**3**
C2M3-7	1	1	**3**	**3**
C2M3-8	1	1	**3**	**3**
C2M3-9	1	1	**3**	**3**
C2M3-10	1	1	4	4

Table 3.8: The number of subclasses found by SDA and SSDA for each class in the C3M3 data sets

Methods / Data sets	SDA			SSDA		
	Class One	Class Two	Class Three	Class One	Class Two	Class Three
C3M3-1	4	4	4	**3**	**3**	**3**
C3M3-2	5	5	5	**3**	**3**	**3**
C3M3-3	**3**	**3**	**3**	**3**	**3**	**3**
C3M3-4	1	1	1	**3**	4	**3**
C3M3-5	1	1	1	**3**	**3**	**3**
C3M3-6	2	2	2	**3**	**3**	**3**
C3M3-7	5	5	5	**3**	**3**	**3**
C3M3-8	1	1	1	**3**	**3**	**3**
C3M3-9	6	6	6	**3**	**3**	**3**
C3M3-10	1	1	1	**3**	**3**	**3**

3.3.4 Q3: How should we utilise the modalities?

After the multiple within-class modalities are found, we need to utilise them in order to reduce dimensionality for the purpose of effective classification. SDA and SSDA provide different solutions, both based on the LDA optimisation process but with different Fisher objectives. To compare these two solutions, we apply the SDA and SSDA optimisation processes to the artificial data sets that consist of within-class modalities (i.e., C2M2, C2M3 and C3M3). In addition, the true number of within-class modalities (True-MN) is used in both SDA and SSDA. The experimental results are presented in Table 3.9, Table 3.10 and Table 3.11.

From Tables 3.9, 3.10 and 3.11, it is clear that the performance of SSDA with True-MN is consistently higher than SDA with True-MN. This suggests that the SSDA optimisation process could better utilise the modalities than the SDA optimisation process. Furthermore, it shows that maximising inter-subclass and inter-class separation at the same time is a worthwhile goal of LDA-like dimensionality reduction when the true modalities are found

in the data.

Table 3.9: The classification accuracy with kNN (k=1) of SDA and SSDA using the true number of within-class modalities on the C2M2 data sets

Methods / Data sets	SDA with True-MN	SSDA with True-MN
C2M2-1	0.9080	**0.9260**
C2M2-2	0.9350	**0.9600**
C2M2-3	0.9670	**0.9750**
C2M2-4	0.9460	**0.9610**
C2M2-5	**0.9230**	**0.9230**
C2M2-6	0.9480	**0.9520**
C2M2-7	0.9380	**0.9570**
C2M2-8	**0.9300**	**0.9300**
C2M2-9	0.8970	**0.9100**
C2M2-10	0.9270	**0.9380**

Table 3.10: The classification accuracy with kNN (k=1) of SDA and SSDA using the true number of within-class modalities on the C2M3 data sets

Methods / Data sets	SDA with True-MN	SSDA with True-MN
C2M3-1	0.9120	**0.9250**
C2M3-2	0.9120	**0.9490**
C2M3-3	0.8940	**0.9220**
C2M3-4	0.9320	**0.9490**
C2M3-5	0.9430	**0.9490**
C2M3-6	0.8920	**0.9190**
C2M3-7	0.8960	**0.9290**
C2M3-8	0.8970	**0.9330**
C2M3-9	0.9000	**0.9200**
C2M3-10	0.8730	**0.8980**

Table 3.11: The classification accuracy with kNN (k=1) of SDA and SSDA using the true number of within-class modalities on the C3M3 data sets

Methods / Data sets	SDA with True-MN	SSDA with True-MN
C3M3-1	0.9093	**0.9433**
C3M3-2	0.9153	**0.9413**
C3M3-3	0.9080	**0.9400**
C3M3-4	0.8580	**0.9167**
C3M3-5	0.9213	**0.9393**
C3M3-6	0.8933	**0.9253**
C3M3-7	0.9047	**0.9280**
C3M3-8	0.9253	**0.9413**
C3M3-9	0.9220	**0.9433**
C3M3-10	0.9480	**0.9520**

3.4 Multimodality in Real Data

Separating within-class multimodalities results in good performance on artificial data when the modality of the data is known. For real-world data, the modality of the data is unknown even if we believe that there should be multimodality, e.g., as in the problem of face recognition discussed in Section 3.1. Can we obtain real benefits by addressing within-class multimodality in real-world data in the same way as for artificial data? This is the question we want to answer in this section. We consider two types of data. One is general data from the UCI data repository [25]; the other is face image data, as it is intuitively plausible that there is within-class multimodality associated with different lighting conditions and different head poses.

In our experiments, we consider k-nearest neighbour (kNN, k=1) as the classifier. We conduct a study on the within-class classification problem by focusing on extracting discriminant features. Some commonly used classifiers have built-in feature

selection/extraction functions. For example, support vector machine and decision tree select features as part of the learning process. The *k*NN classifier does not have any built-in feature selection/extraction function, so it is selected and used in our experiments. Additionally, we use ten-fold cross-validation as the evaluation framework, and *Estimated Mean Accuracy* (EMA) and *Standard Error of the Mean* (SEM) [49] as the evaluation metrics: $EMA = \frac{\sum_{i=1}^{10} p_i}{10}$, where p_i denotes the percentage of correct classification in the ith fold validation; $SEM = \frac{\delta}{\sqrt{10}}$, where $\delta = \sqrt{\frac{\sum_{i=1}^{10}(p_i - EMA)^2}{9}}$. So, the higher EMA and lower SEM, the better is classification performance. Moreover, to make the evaluation results more reliable, we ran each experiment 10 times using ten-fold cross-validation, and reported the average EMA (AEMA) and average SEM (ASEM).

3.4.1 General Data

We select eleven UCI data sets using two criteria: (1) all attributes must be numerical; (2) there must be many attributes so that dimensionality reduction is meaningful. General information about the eleven UCI data sets is shown in Table 3.12.

Furthermore, we compare SSDA and SDA against adaptive local linear discriminant analysis (ALLDA) [93]. To compare ALLDA as fairly as possible, we follow the experimental settings used in [93] since we do not have the source code of ALLDA. In [93], four UCI data sets are used to test the performance of ALLDA. They are Australian, Heart, Pima and Diabetes, respectively. We could not find the Diabetes data set corresponding to the description in the [93], so we compare SDA and SSDA with ALLDA on the remaining three data sets. The experimental settings used in [93] are: (1) several samples are randomly selected from every class, with the same proportion used as training data and the rest of the samples used as testing data. The splits of Australian, Heart and Pima data sets are described in Table 3.13; (2)1-nearest neighbour is used as the classifier, and each experiment is conducted using 20 random splits; (3) the mean accuracy (Macc) and standard deviation (Std) are the evaluation metrics for classification performance.

Table 3.12: General information about the eleven UCI data sets used, where #I denotes the number of instances, #C denotes the number of classes and #A denotes the number of attributes

Name of data set (Acronym)	#I	#C	#A
QSAR Biolodegradation (QSAR-B)	1055	2	41
Climate Model Simulation Crashes (CMSC)	540	2	18
Diabetic Retinopathy (DR)	1151	2	19
Multiple Feature-fou (MF-fou)	2000	10	76
Musk(Version 1)-Clearn1 (M1-C1)	476	2	166
Parkinsons	195	2	22
Statlog Project (SP)	846	4	18
White Wine Quality (WWQ)	4898	7	11
Yeast	1484	10	8
Isolet	7797	26	617
Vertebral	310	2	6

Table 3.13: General information and the split about Australian, Heart and Pima data set, where #C denotes the number of classes, #Training denotes the number of training samples, #Testing denotes the number of testing samples and #A denotes the number of attributes

Name of data set	#C	#Training	#Testing	#A
Australian	2	207	483	14
Heart	2	54	216	13
Pima	2	149	619	8

Table 3.14: AEMA±ASEM values with kNN (k=1) of Original, LDA, SDA and SSDA on Eleven UCI data sets

Methods / Data sets	Original	LDA	SDA	SSDA
	AEMA ± ASEM	AEMA ± ASEM	AEMA ± ASEM	AEMA ± ASEM
QSAR-B	0.7928 ± 0.0139	0.7954 ± 0.0111	0.7580 ± 0.0132	**0.8381 ± 0.0102**
CMSC	0.8895 ± 0.0106	0.9384 ± 0.0087	0.9389 ± 0.0074	**0.9454 ± 0.0093**
DR	0.6172 ± 0.0129	0.6448 ± 0.0138	0.6451 ± 0.0148	**0.6796 ± 0.0122**
MF-fou	0.8269 ± 0.0068	0.8152 ± 0.0071	**0.8374 ± 0.0067**	0.8343 ± 0.0063
M1-C1	0.8578 ± 0.0142	0.7881 ± 0.0193	0.7462 ± 0.0219	**0.8814 ± 0.0149**
Parkinsons	0.8454 ± 0.0254	0.8389 ± 0.0219	0.8424 ± 0.0255	**0.8616 ± 0.0213**
SP	0.7020 ± 0.0136	0.7879 ± 0.0116	0.7744 ± 0.0122	**0.8313 ± 0.0105**
WWQ	0.5980 ± 0.0058	0.6254 ± 0.0062	0.6096 ± 0.0066	**0.6339 ± 0.0080**
Yeast	0.5238 ± 0.0152	0.5217 ± 0.0137	0.5295 ± 0.0133	**0.5328 ± 0.0148**
Isolet	0.8967 ± 0.0030	0.9469 ± 0.0260	0.9488 ± 0.0025	**0.9594 ± 0.0022**
Vertebral	**0.8390 ± 0.0188**	0.7742 ± 0.0203	0.8265 ± 0.0198	0.8119 ± 0.0229

Table 3.15: Macc±Std values with kNN (k=1) of Original, LDA, SDA, SSDA and ALLDA on Australian, Heart and Pima data set, where the results of ALLDA are cited from [93]

Methods / Data sets	Original	LDA	SDA	SSDA	ALLDA
	Macc±Std	Macc±Std	Macc±Std	Macc±Std	Macc±Std
Australian	0.6340±0.0163	0.8000±0.0247	0.6262±0.0184	**0.8148±0.0164**	0.7775±0.0198
Heart	0.6116±0.0279	0.7750±0.0328	0.7303±0.0254	**0.7887±0.0207**	0.7431±0.0064
Pima	0.6670±0.0167	0.6845±0.0181	**0.6945±0.0187**	0.6933±0.0162	0.6763±0.0249

Experimental results are presented in Table 3.14 and Table 3.15. The experimental results of ALLDA in Table 3.15 are cited from [93]. From these results we note the following observations:

- LDA, SDA and SSDA achieve better performance than Original on the majority of the UCI data sets. This further verifies the conclusion drawn by using artificial data sets that it is necessary to deal with the issue of within-class multimodality.

- From Table 3.14, it is clear that SSDA achieves better classification performance than Original and SDA on 10 out of 11 data sets. Moreover, SSDA outperforms LDA on all UCI data sets. However, it is interesting to see that Original obtains the best classification accuracy on the Vertebral data set. This could result from the few number of features in this data set, which makes dimensionality reduction meaningless.

- Compared with Original and LDA, both SDA and SSDA have superior performance on CMSC, DR, MF-fou, Parkinsons, Yeast and Isolet. This suggests that these data sets are likely to have salient within-class multimodalities. Figure 3.4 provides a visualisation of these data sets in a two-dimensional space by t-SNE [80], where different colours represent different classes. t-SNE is a technique for visualising high-dimensional data sets by giving each sample a location in a two- or three-dimensional space. It can be observed that these data sets comprise different classes and some class clusters consist of several clusters, which correspond to

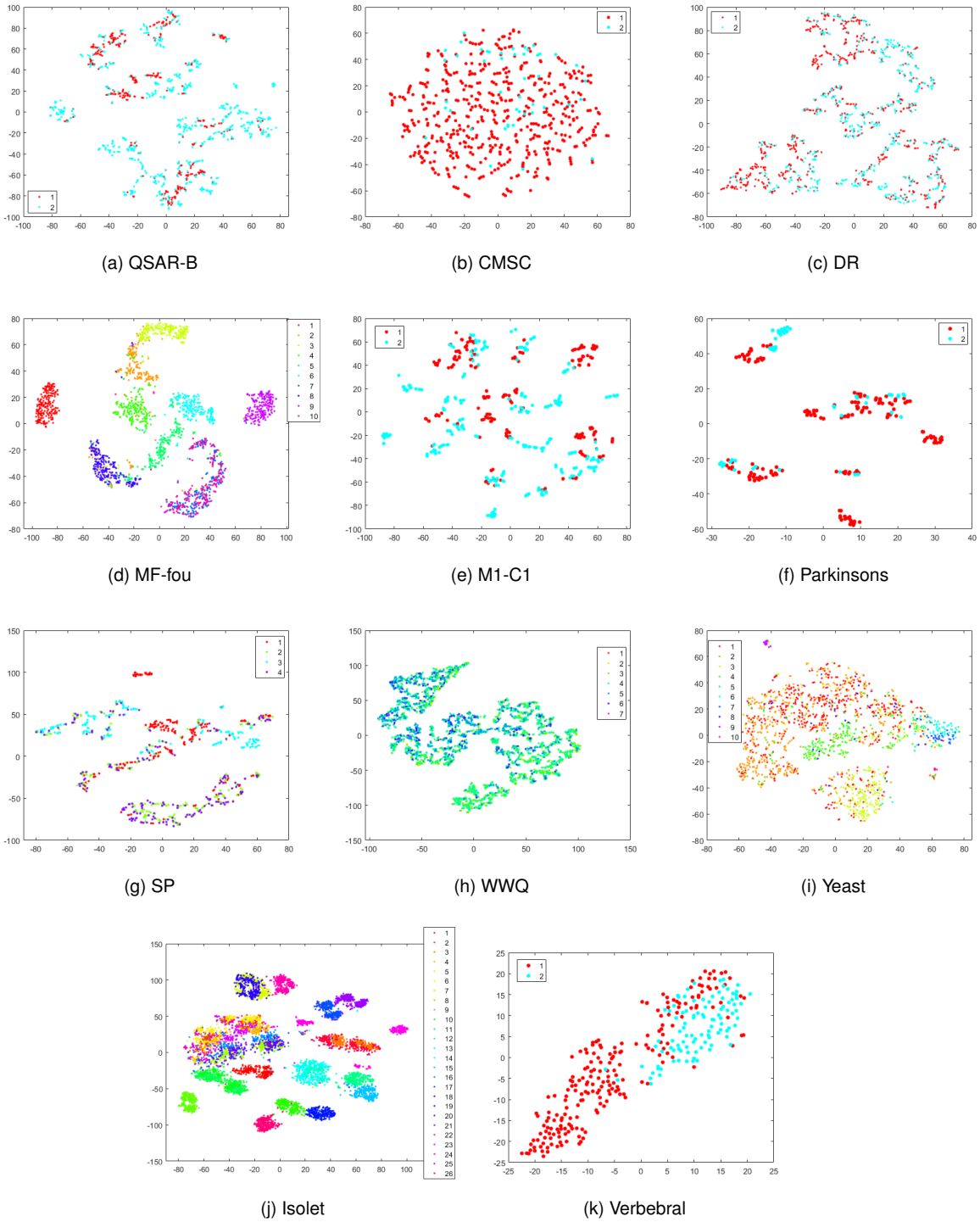
within-class modalities. In particular, the presence of multimodality is clear in Parkinsons, where Class 1 consists of several red clusters and Class 2 consists of several cyan clusters.

3.4.2 Face Image Data

We conduct face recognition experiments on two widely used face databases: AR face database [82] and FERET face database [98]. Face Recognition is a multi-class classification problem, where each person is regarded as a class. Face recognition attempts to determine whether a face image is from someone in the database when we have a collection of images for each person in the database. A person's set of face images may contain multiple modalities when they are captured in different illumination conditions or head poses. So, the purpose of this study is to test whether the within-class multimodality methods discussed can bring benefit to this problem.

In our experiments, the images are represented using their pixel values, resulting in a large numbers of features. Therefore, our face recognition task becomes a *small sample size* (SSS) problem [108]. To deal with this problem, a two stage PCA + LDA method [7] is used. We use PCA to reduce data dimensionality, retaining principal components that can explain 90% of the variance, before LDA, SDA and SSDA are used. Details of the two face databases used in our experiments are given below:

- *AR face database*: The AR face database contains frontal-view face images of 126 different persons (70 males and 56 females). Each person was photographed under different lighting conditions (normal lighting, normal lighting and left light on, normal lighting and right light on, normal lighting and both left and right lights on) and distinct facial expressions (neutral expression, smile, anger, and scream), and some images have partial occlusions (sunglasses or scarf). For each person, a total of 13 images were taken in each session for a total of two sessions, which were separated by an interval of two weeks. Hence, there are 26 frontal face images per



(a) QSAR-B (b) CMSC (c) DR

(d) MF-fou (e) M1-C1 (f) Parkinsons

(g) SP (h) WWQ (i) Yeast

(j) Isolet (k) Verbebral

Figure 3.4: The data visualisation of QSAR-B, CMSC, DR, MF-fou, M1-C1, Parkinsons, WWQ, SP, Yeast, Isolet and Vertebral in a two-dimensional space.

(a) Examples of images in the AR face database

(b) Examples of images in the FERET face database

Figure 3.5: Sample images from the face databases.

person. In our experiments, we use a subset of the AR face data set, which comprises 700 face images from 100 persons. We use 7 non-occluded face images of each person taken under different lighting conditions and different facial expressions from the first session. Further, we crop the face part of the image and then resize all images to a standard image size of 80 x 100 pixels (see Figure 3.5(a) for some examples). Thus, every face image in the AR database has 8000 features.

- *FERET face database*: The FERET face database includes over 10,000 face images, which have different head poses, lighting conditions and expressions. In our experiments, we use a subset of the FERET face database that consists of 700 images from 100 people, with 7 images per person. Again the face portion of each image is cropped out and normalised to a standard image size of 100 x 100 pixels (see Figure 3.5(b) for some examples). So, we have 10000 features for each image of FERET.

We run experiments with Original, LDA, SDA and SSDA on the AR and FERET face databases 10 times using ten-fold cross-validation. Experimental results are shown in Table 3.16 and Table 3.17. It is clear that SSDA achieves higher face recognition accuracy than the other three methods on both face image databases; SDA also outperforms Original and LDA on both face databases. These results suggest that within-class multimodality does exist in these image databases, and tackling within-class multimodality in the manner of SDA and SSDA does bring benefits.

Table 3.16: EMA ±SEM values with *k*NN (*k*=1) of Original,LDA, SDA and SSDA on the AR face database

Methods / AR	Original	LDA	SDA	SSDA
	EMA ± SEM	EMA ± SEM	EMA ± SEM	EMA ± SEM
1	0.5099 ± 0.0158	0.5978 ± 0.0195	0.7806 ± 0.0249	**0.8397 ± 0.0076**
2	0.5107 ± 0.0117	0.5706 ± 0.0339	0.7822 ± 0.0276	**0.8511 ± 0.0182**
3	0.5092 ± 0.0173	0.5866 ± 0.0187	0.7188 ± 0.0352	**0.8431 ± 0.0159**
4	0.5081 ± 0.0195	0.5647 ± 0.0250	0.8052 ± 0.0220	**0.8356 ± 0.0195**
5	0.5068 ± 0.0203	0.5877 ± 0.0225	0.7682 ± 0.0393	**0.8517 ± 0.0156**
6	0.5129 ± 0.0186	0.5912 ± 0.0237	0.7814 ± 0.0317	**0.8432 ± 0.0129**
7	0.5128 ± 0.0168	0.5761 ± 0.0218	0.7366 ± 0.0240	**0.8309 ± 0.0103**
8	0.5136 ± 0.0132	0.5716 ± 0.0221	0.7402 ± 0.0307	**0.8326 ± 0.0208**
9	0.5115 ± 0.0162	0.5770 ± 0.0270	0.7830 ± 0.0085	**0.8539 ± 0.0127**
10	0.5088 ± 0.0183	0.5636 ± 0.0285	0.7939 ± 0.0183	**0.8459 ± 0.0125**
Average	0.5104 ± 0.0168	0.5787 ± 0.0243	0.7690 ± 0.0262	**0.8428 ± 0.0146**

Table 3.17: EMA ±SEM values with *k*NN (*k*=1) of Original,LDA, SDA and SSDA on the FERET face database

Methods / FERET	Original	LDA	SDA	SSDA
	EMA ± SEM	EMA ± SEM	EMA ± SEM	EMA ± SEM
1	0.5381 ± 0.0109	0.6065 ± 0.0158	0.6131 ± 0.0179	**0.6844 ± 0.0110**
2	0.5465 ± 0.0167	0.5646 ± 0.0187	0.6059 ± 0.0196	**0.6912 ± 0.0192**
3	0.5375 ± 0.0147	0.5866 ± 0.0161	0.5961 ± 0.0218	**0.6836 ± 0.0199**
4	0.5328 ± 0.0155	0.5898 ± 0.0172	0.6172 ± 0.0153	**0.6798 ± 0.0147**
5	0.5394 ± 0.0190	0.5979 ± 0.0147	0.6118 ± 0.0234	**0.6902 ± 0.0102**
6	0.5425 ± 0.0163	0.6038 ± 0.0192	0.6130 ± 0.0221	**0.7095 ± 0.0179**
7	0.5349 ± 0.0250	0.5867 ± 0.0217	0.6065 ± 0.0160	**0.6884 ± 0.0229**
8	0.5340 ± 0.0240	0.5913 ± 0.0179	0.6061 ± 0.0166	**0.6747 ± 0.0187**
9	0.5400 ± 0.0255	0.5961 ± 0.0191	0.6071 ± 0.0185	**0.6979 ± 0.0197**
10	0.5311 ± 0.0174	0.5710 ± 0.0212	0.6025 ± 0.0223	**0.6868 ± 0.0187**
Average	0.5377 ± 0.0185	0.5894 ± 0.0182	0.6079 ± 0.0193	**0.6887 ± 0.0173**

Furthermore, we want to see what within-class modalities SDA and SSDA can find for AR and FERET, and if the modalities found are consistent with reality. To achieve this, we apply SDA and SSDA to all images of AR and FERET, respectively. Therefore, the maximum number of modalities for each class is set as 7 for both methods, since every person only has 7 images in AR and FERET databases. According to the within-modalities found by SDA and SSDA shown in Figure 3.6 and Figure 3.7, we obtain the following observations:

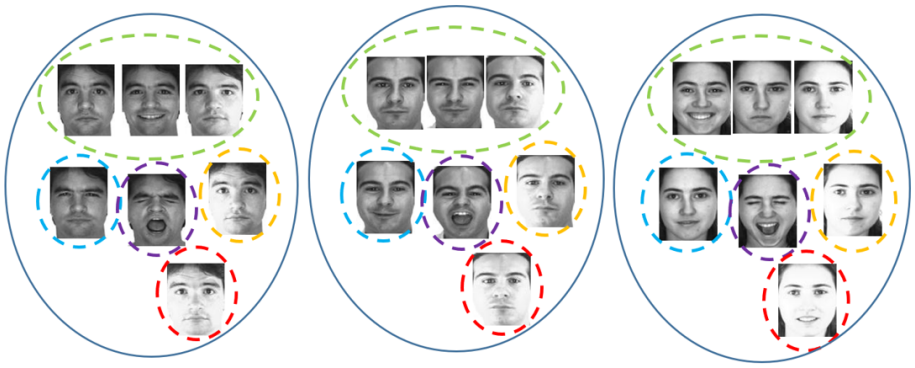
- From Figure 3.6, it is readily seen that the four modalities found by SSDA correspond to four different illumination conditions existing in the AR database: normal lighting, normal lighting and left light on, normal lighting and right light on, normal lighting and both left and right light on. Although SDA successfully finds two types of illumination modalities: normal lighting and left light on, normal lighting and both left and right light on, it mixes up the images with normal lighting and left light on.

- For the FERET database, both SDA and SSDA find different types of within-class modalities for different classes, as shown in Figure 3.7. Again, SSDA identifies two types of illumination modalities for each class: normal lighting and low lighting, but SDA fails to find the modality with low lighting for some classes, such as Figure 3.7(a)(2).

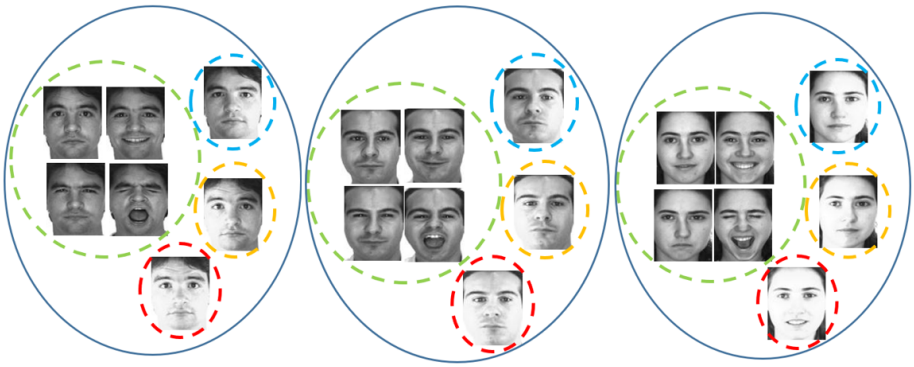
- Apart from identifying the illumination modalities in the FERET database, SSDA can find all correct head pose modalities for some classes (see Figure 3.7(b)(3)): frontal modality, leftwards modalities with two different angles and rightwards modalities with two different angles. In addition, SDA also can find some correct head pose modalities for some classes, for example, the modalities represented by the cyan and purple dotted circles shown in Figure 3.7(a)(3).

Therefore, all results from these experiments on two real face databases are consistent

with the results on the artificial data sets. When there is within-class multimodality in the data, dealing with the multimodality problem in the manner of either SDA or SSDA is beneficial, and, furthermore, the SSDA approach is better than the SDA approach. Interestingly, we have shown that SDA and SSDA offer potential solutions to a challenging problem – face recognition under different lighting and head pose conditions.

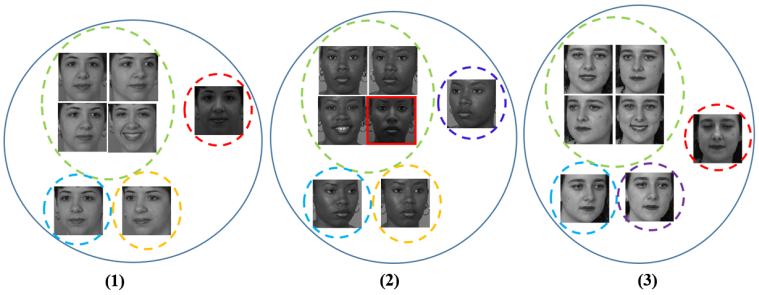


(a) Modality distributions found by SDA

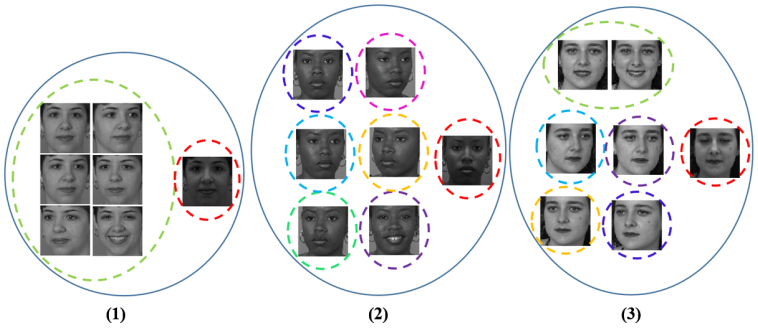


(b) Modality distributions found by SSDA

Figure 3.6: Examples of modality distributions found by SDA and SSDA on the AR face database, where dotted circles with different colours represent different modalities found by SDA and SSDA. In (b), the green dotted circle represents the illumination modality with normal lighting; the cyan dotted circle represents the illumination modality with normal lighting and right light on; the orange dotted circle represents the illumination modality with normal lighting and left light on; the red dotted circle represents the illumination modality with normal lighting and both left and right light on.



(a) Modality distributions found by SDA



(b) Modality distributions found by SSDA

Figure 3.7: Examples of modality distributions found by SDA and SSDA on the FERET face database, where dotted circles with different colours represent different modalities found by SDA and SSDA.

Table 3.18: Running time, in seconds, of Original, LDA, SDA and SSDA on eleven UCI data sets and two face databases 10 times using ten-fold cross-validation

Methods / Data sets	Original	LDA	SDA	SSDA
QSAR-B	1.7328	2.6739	20.6197	9.7620
CMSC	0.6416	1.0664	6.7935	2.4880
DR	0.7180	1.2667	17.6121	12.0517
MF-fou	1.7049	3.4528	42.9585	10.1369
M1-C1	0.8752	4.4181	17.4592	12.0827
Parkinsons	0.6188	0.9393	8.4054	1.8269
SP	0.6151	1.2622	17.6392	4.5284
WWQ	1.9772	2.8096	87.9767	117.6188
Yeast	0.9834	1.6027	16.4443	9.5505
Isolet	134.1505	1057.5846	1951.6677	1140.4812
Verbebral	0.6817	1.0112	3.8639	3.7453
AR	34.3839	19.2355	63.6227	42.9422
FERET	43.3403	21.9655	306.4279	65.8793

3.4.3 The Results: Runtime Performance

Runtime results of Original, LDA, SDA and SSDA are shown in Table 3.18. It is clear that SSDA is slower than Original and LDA, but faster than SDA in most of the data sets.

3.5 Summary

Within-class multimodality exists in real-world data and was first studied by [137] and more recently by [119], but many questions are unanswered about within-class multimodality, and its true value is not uncovered fully. This chapter presents an extensive study of the within-class multimodality problem through experiments on both artificial data and real data in order to establish a strong case for *within-class multimodal classification*.

It has been shown using both artificial data and real data that when within-class multimodality is present, maximising between-subclass separation, between-class separation and within-class compactness at the same time in the manner of SDA or SSDA increases classification performance, with SSDA being the better approach. It is also shown that addressing within-class multimodality this way is optimal if the true number of modalities is known. Interestingly, the experiment on face image databases suggests that SDA and SSDA offer an alternative approach to addressing face recognition under different lighting and head pose conditions.

We believe that a strong case for within-class multimodal classification has been established. We also believe that this classification approach offers a new perspective on improving existing classification algorithms such as Gaussian mixture model and convolutional neural networks, and even devising new classification algorithms.

CHAPTER IV

Global Subclass Discriminant Analysis

In the previous chapter, it was found that when within-class multimodality is present, the concurrent maximisation of between-class separation, between-subclass separation and within-class compactness can lead to significant classification performance gains. Based on this, in this chapter, we propose a novel feature extraction method, called *global subclass discriminant analysis* (GSDA), which is a variant of subclass-based LDA.

4.1 Motivation

The goal of LDA is to find a subspace where the interclass distance is maximised and simultaneously the intraclass distance is minimised. Analytic solutions to this optimisation problem exist under the assumption that all classes of data have equal covariance matrices. This assumption implies that the data are linearly separable [131] so the analytic solutions are indeed solutions to the LDA optimisation problem when this assumption is true, possible solutions when the data are linearly separable, but definitely not solutions when the data are not linearly separable.

In real world applications, however, it is possible that the classes of data have different covariance matrices or that the classes are not linearly separable. Thus, LDA may suffer from dramatic performance degradation. We call this the *nonlinear data problem*.

To address this issue, Mika et al.[86] proposed kernel discriminant analysis (KDA),

which is a nonlinear generalisation of LDA for binary classification based on the kernel trick, the commonly used technique to design nonlinear classifiers from linear ones. The kernel trick was originally used in support vector machines (SVM) [14] and kernel principal component analysis (PCA) [104]. The idea of the kernel trick is to use Mercer kernels (e.g. a Gaussian kernel) to implicitly map nonlinear data to a high-dimensional space $\mathcal{F}$ where the classes of data become linearly separable. In [86], KDA employs a Gaussian kernel and polynomial kernel of degree two to implicitly map the original data into the space $\mathcal{F}$. Then, in the space $\mathcal{F}$, KDA defines a between-class scatter matrix and a regularised within-class scatter matrix as the measurements of interclass distance and intraclass distance, respectively. Finally, KDA extracts discriminant features by using the LDA optimisation process. KDA is better than LDA on nonlinear data, and so many variants of KDA have been proposed [97, 113, 53]. However, there is a problem with KDA and its variants. There is no kernel function that works well for all data sets, so it is necessary to identify a suitable kernel for each specific data set before KDA is applied. There are, however, no general guidelines for doing this, so trial and error has to be used to identify a suitable kernel for a specific data set.

Subclass-based methods are a class of alternative solutions to the optimisation problem of LDA, which do not rely on the assumption that all classes have equal covariance matrices. The main idea of subclass-based methods is to partition a class into several subclasses, and seek to maximise interclass distance and minimise intraclass distance based on subclasses. As a result, nonlinear data become linearly separable at the subclass-level. Take the two classes in Figure 4.1 as an example. It is clear that Class One and Class Two cannot be separated linearly at the class-level, but the four subclasses are linearly separable.

Zhu and Martinez [137] proposed *subclass discriminant analysis* (SDA), a variant of LDA, to solve the nonlinear data problem. SDA utilises a nearest neighbour-based clustering algorithm and a stability criterion to partition every class into the same number

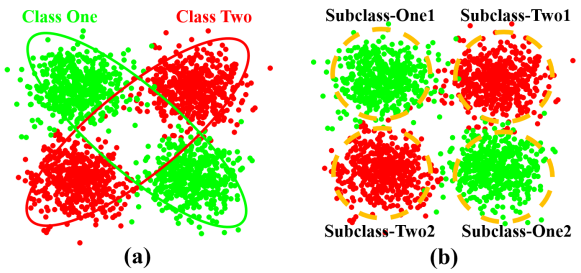


Figure 4.1: Linear separation at class and subclass levels. (a) A data set with two classes, in green and red respectively, which can not be linearly separated. (b) The same data set with classes partitioned into subclasses: subclass-One1 and subclass-One2 in Class One, subclass-Two1 and subclass-Two2 in Class Two. Each subclass is represented by an orange dashed circle.

of subclasses. SDA measures interclass distance by using a between-subclass scatter matrix, and intraclass distance by using a sample covariance matrix. Finally, SDA finds the subspace that maximises interclass distance and minimises intraclass distance through the LDA optimisation mechanism.

Mixture subclass discriminant analysis (MSDA) [38] is another variant of LDA of the same kind. MSDA partitions a class into subclasses only when this class does not have a Gaussian distribution according to the non-Gaussianity criterion the authors proposed, where the number of subclasses is determined according to the same stability criterion as in SDA [137]. As a result, different classes may have different numbers of subclasses. MSDA uses the same between-subclass scatter matrix as in SDA and a new within-subclass scatter matrix to measure the intraclass distance.

Unlike SDA and MSDA, *separability-oriented subclass discriminant analysis* (SSDA) [119] employs a separability criterion to partition every class into a number of non-overlapping subclasses. Based on these non-overlapping subclasses, SSDA defines a new between-subclass scatter matrix and uses the LDA optimisation mechanism to find a subspace that simultaneously maximises between-class distance and between-subclass distance, and minimises within-class distance.

We note that all of the proposed subclass-based methods restrict themselves to finding subclasses within a class. Although they can find a subspace to linearly separate data at

the subclass level, they select subclasses within every class separately, i.e., locally. Consequently, some subclasses may not form clusters within a class so they may not be found. Consequently they are not sufficiently separated from other classes or subclasses, resulting in missed opportunities and hence reduction in performance. We call this the *local separation* problem, see Figure 4.2 for an illustration of this problem. The two classes, in red and green, have substantial overlap. If we partition Class One based on only Class One data (in green), we will get two local clusters (LC), LC1 and LC2 (Figure 4.2(b), top), which are taken as two subclasses of Class One. Similarly, if we partition Class Two based on only Class Two data (in red), we will get two local clusters, LC3 and LC4 (Figure 4.2(b), bottom), which are taken as two subclasses of Class Two. However, if we partition the whole data set we will get five global clusters (GC), see GC1-GC5 in Figure 4.2(a). When we look at the classes separately, we have four global clusters for Class One (GC2-1, GC3, GC4, GC5-1) and three global clusters for Class Two (GC1, GC2-2, GC5-2). Note that GC2-1 and GC2-2 (also GC5-1 and GC5-2) are substantially overlapping, and they belong to different classes. So, if we take them as separate subclasses, then a subclass based method would push them apart (i.e. separate them) as much as possible, resulting in higher classification performance. This motivated us to search for a new method which clusters the whole data to find subclasses of every class before we apply the LDA mechanism. This effort results in *Global Subclass Discriminant Analysis* (GSDA), the subject of this chapter.

4.2 Global Subclass Discriminant Analysis

Similar to SDA, MSDA and SSDA, GSDA also uses the idea of subclass. However, SDA, MSDA and SSDA generate subclasses by clustering each class into clusters without considering other classes and taking these clusters as subclasses, so these clusters and subclasses are *local*. In contrast, GSDA generates subclasses by clustering the whole data into clusters, each of which is separated into possibly multiple subclasses, one per class,

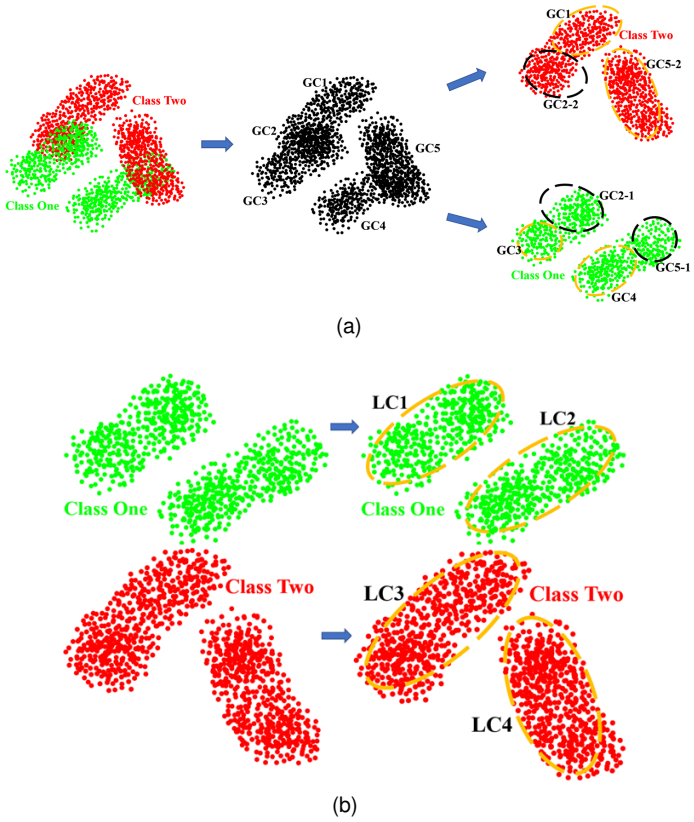


Figure 4.2: Class partition at global and local levels. Class one and Class two are shown in green and red, respectively, while the samples shown in black describe the global clusters GC1-GC5. (a) Global class partition: resulting in four and three global clusters (GC) in Class One and Class Two , respectively. (b) Local class partition: resulting in two local clusters (LC) in each class.

so these clusters and subclasses are *global*. If a global cluster consists of data from different classes, it is a *boundary cluster*. It is well known that boundary data are hard to separate by their class memberships, and the performance of a classifier is to a large extent dependent on how it handles boundary data [43]. GSDA seeks to separate boundary data explicitly by their class memberships using the LDA mechanism in order to improve classification performance. GSDA takes a global cluster that contains data from multiple classes as multiple subclasses, one for each class, which are called *global subclasses* or G-subclasses for short. Similarly, subclasses in SDA/MSDA/SSDA are called *local subclasses* or L-subclasses for short.

We propose an efficient clustering method for GSDA, *rough-refine clustering*, for

finding G-subclasses. The goal of GSDA is then to maximise the separation of the G-subclasses and the compactness of all G-subclasses. In the rest of this section we present details of the *rough-refine clustering* method and our way of composing the Fisher-Rao's criterion.

4.2.1 Rough-Refine Clustering

We need a clustering algorithm with the following qualities: (1) it should be fast, as it is used as part of our discriminant analysis; (2) it should be able to automatically determine the number of clusters. Off-the-shelf clustering algorithms mostly require the number of clusters to be given *a priori*. Some clustering algorithms can determine this number automatically, but are generally quite complicated. Therefore, we design a new algorithm to find G-subclasses, *rough-refine clustering*, which is a mixture of clustering and post-processing.

The rough-refine clustering method consists of two parts: rough clustering and refine clustering. Rough clustering clusters the whole data set irrespective of their class memberships, resulting in *global clusters*, while refine clustering creates class-specific clusters from the global clusters. We use the HC-SC clustering strategy [119] for rough clustering, which applies hierarchical clustering and stops according to the separability criterion, resulting in an 'optimal' set of global clusters.

In refine clustering, a global cluster GC that contains data from different classes is separated into different sets $GC_i = \{\mathbf{x} \in GC : \mathbf{x} \text{ belongs to class } i\}$, one for each class i. Each GC_i is taken as one G-subclass for class i. If GC contains only data from one class i, then GC is taken as a G-subclass for class i.

The rough-refine clustering process is summarised in Algorithm 2. The algorithm is illustrated by an example in Figure 4.3. It is clear from this example that the number of G-subclasses obtained by the rough-refine clustering algorithm varies for different classes. This is different to SDA, which requires the same number of L-subclasses be found for

every class. This generalisation is important as there is nothing to guarantee that every class comprises the same number of subclasses/clusters/distributions (i.e. Gaussian distributions).

Algorithm 2 Rough-refine clustering.

Input: T_{set} and k_{max}. $T_{set} = \{s_{ij}|i = 1, 2, ..., n; j = 1, 2, ..., n_i\}$ denotes the training set, where s_{ij} represents the jth sample of the ith class, n is the class number and n_i is the sample number of the ith class. k_{max} is the maximum number of subclasses, which is a parameter in HC-SC clustering method.

Output: $Gsubclass$. $Gsubclass = \{b_{ikm}|i = 1, 2, ..., n; k = 1, 2, ..., k_i; m = 1, 2, ..., m_{ik}\}$ denotes the subclass set, where b_{ikm} represents the mth sample of the kth subclass in the ith class, k_i is the subclass number in the ith class and m_{ik} is the sample number of the kth subclass in the ith class.

Apply HC-SC on T_{set} with k_{max} to obtain cluster labels – $CluLabel = \{l_{ij}|i = 1, 2, .., n; j = 1, 2, ..., n_i\}$, where l_{ij} represents the cluster label of the jth sample in the ith class.
$CluLabel$ = HC-SC(T_{set}, k_{max});
$Gsubclass = \{\}$;
for $i = 1$ to n **do**
    k = 1; k_i = 0; m_{ik} = 0; $subclass = \{\}$;
    for $j = 1$ to n_i **do**
        if the cluster label of any element in $subclass$ equals l_{ij} **then**
            return the subclass label of the first found element to k;
            m_{ik} = m_{ik} + 1;
            $b_{ikm_{ik}}$ = s_{ij};
            $subclass = subclass \cup \{b_{ikm_{ik}}\}$;
        else
            k_i = k_i + 1; m_{ik_i} = 1;
            $b_{ik_im_{ik_i}}$ = s_{ij};
            $subclass = subclass \cup \{b_{ik_im_{ik_i}}\}$;
        end if
    end for
    $Gsubclass = Gsubclass \cup subclass$;
end for
return $Gsubclass$;

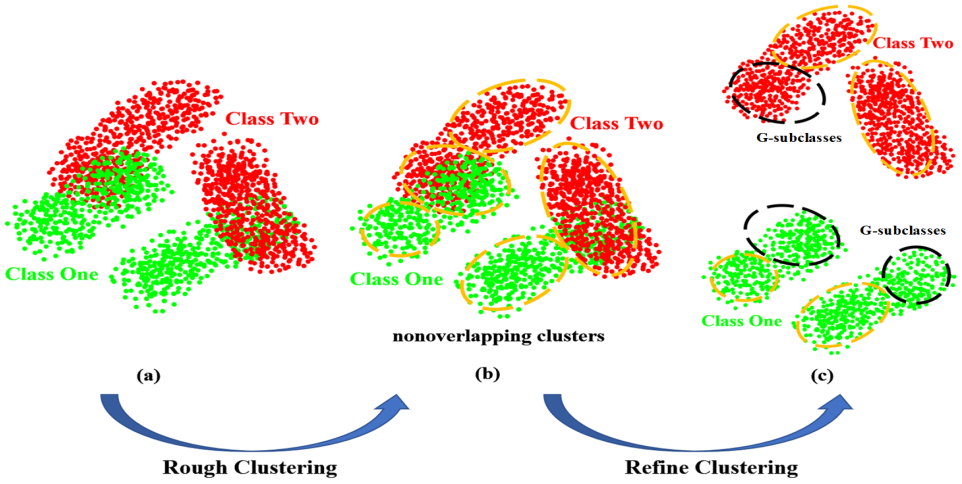


Figure 4.3: An illustration of how the rough-refine algorithm finds G-subclasses. (a) A data set in which there are two classes in green and red. (b) Five non-overlapping global clusters as a result of rough clustering, denoted by dashed orange circles. (c) G-subclasses denoted by dashed orange and black circles: 4 in Class One and 3 in Class Two, as a result of refine clustering.

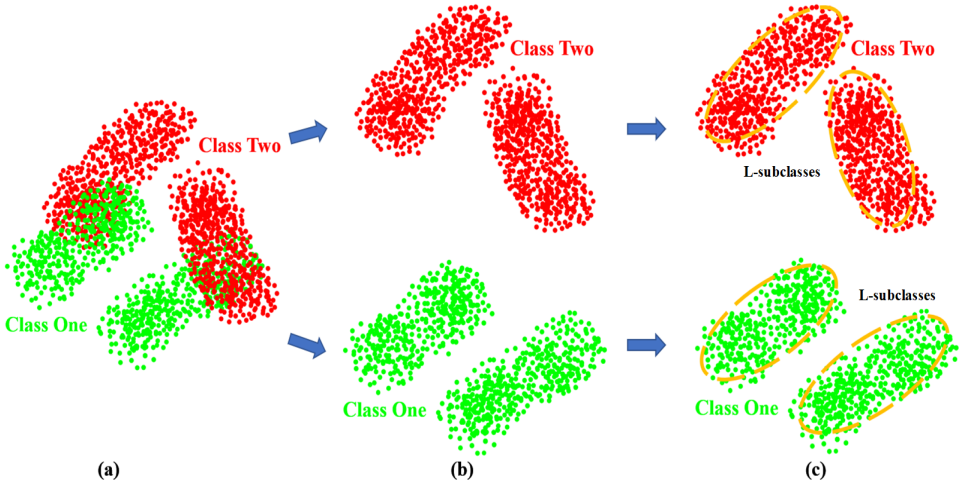


Figure 4.4: An illustration of how to get L-subclasses. (a) The same data set as in Figure 4.3(a), in which there are two classes in green and red, and each class consists of two clusters. (b) Clusters in each of the two classes separately. (c) L-subclasses denoted by the dashed orange circles, each corresponding to a cluster in (b).

Figure 4.4 illustrates a local separation process on the same data but uses the clustering method only on class-level to produce L-subclasses. Comparing the G-subclasses in Figure

4.3(c) with L-subclasses in Figure 4.4(c), it is clear that our global approach, through the rough-refine clustering algorithm, can identify not only more subclasses but also **boundary** subclasses at the intersection of different classes. For example, the dashed black circles in Figure 4.3(c) are boundary subclasses. It is well known that boundary data are notoriously hard to separate correctly, and are usually the culprits for incorrect classifications.

Once G-subclasses are identified, we seek to separate them in order to maximise the distance between these G-subclasses and minimise the distance within these G-subclasses. This is achieved through the LDA optimisation process with new scatter matrices, which are described in the next subsection.

4.2.2 The Re-defined Fisher-Rao's Criterion

A key component of the LDA optimisation is the Fisher-Rao's Criterion:

$$J(\mathbf{W}) = \frac{tr(\mathbf{WAW}^T)}{tr(\mathbf{WBW}^T)},$$

where matrices $\mathbf{A}$ and $\mathbf{B}$ can be defined for different purposes. In GSDA, $\mathbf{A}$ is taken to be a scatter matrix between different G-subclasses, $\mathbf{S_{bGsb}}$, and $\mathbf{B}$ is taken to be a scatter matrix within G-subclasses, $\mathbf{S_{wGsb}}$. These two matrices are defined below:

$$\mathbf{S_{bGsb}} = \sum_{k=1}^{K} (\mu_k - \mu)(\mu_k - \mu)^T, \tag{4.1}$$

$$\mathbf{S_{wGsb}} = \frac{1}{N} \sum_{k=1}^{K} \sum_{l=1}^{N_k} (\mathbf{x}_{kl} - \mu_k)(\mathbf{x}_{kl} - \mu_k)^T, \tag{4.2}$$

where K is the total number of G-subclasses in a data set, μ_k is the mean of G-subclass k and μ is the global mean of the data set, N_k is the number of samples in G-subclass k and $\mathbf{x}_{kl}$ is the lth sample in G-subclass k.

The Fisher-Rao's criterion, i.e. the objective function of GSDA, is re-defined as:

$$J(\mathbf{W})^{GSDA} = \frac{tr(\mathbf{W}\mathbf{S_{bGsb}}\mathbf{W}^T)}{tr(\mathbf{W}\mathbf{S_{wGsb}}\mathbf{W}^T)}. \tag{4.3}$$

The matrix $\mathbf{W}^*$ that maximises the GSDA objective is obtained by solving the generalised eigenvalue decomposition equation

$$\mathbf{S_{wGsb}}^{-1}\mathbf{S_{bGsb}}\mathbf{W}^* = \mathbf{W}^*\Lambda.$$

$\mathbf{W}^*$ is the sought-after transformation matrix, which transforms data from the original space to GSDA's LDA space, or simply the GSDA space, which is spanned by the eigenvectors of matrix $\mathbf{S_{wGsb}}^{-1}\mathbf{S_{bGsb}}$.

According to the definition of $\mathbf{S_{bGsb}}$ and $\mathbf{S_{wGsb}}$ in Equations (4.1) and (4.2), GSDA aims to maximise between-class separation and within-class compactness at the subclass level rather than the class level. If this is achieved, separation and compactness should also be optimal at the class level. Additionally, instead of trying only to compact G-subclasses, GSDA also tries to separate G-subclasses.

4.3 Evaluation Using Artificial Data

To evaluate GSDA, we first use an artificial data set. We compare GSDA with SDA/MSDA/SSDA by visualising data in the original data space as well as the feature spaces by different methods, and by measuring the separability of the transformed data in different ways. For clarity, we visualise data in two-dimensional space. Since the data are high dimensional, we use the T-distributed Stochastic Neighbour Embedding (t-SNE) algorithm [80] to reduce dimensions. t-SNE is a nonlinear dimensionality reduction technique and is widely used to visualise high-dimensional data in a low-dimensional space of two or three dimensions.

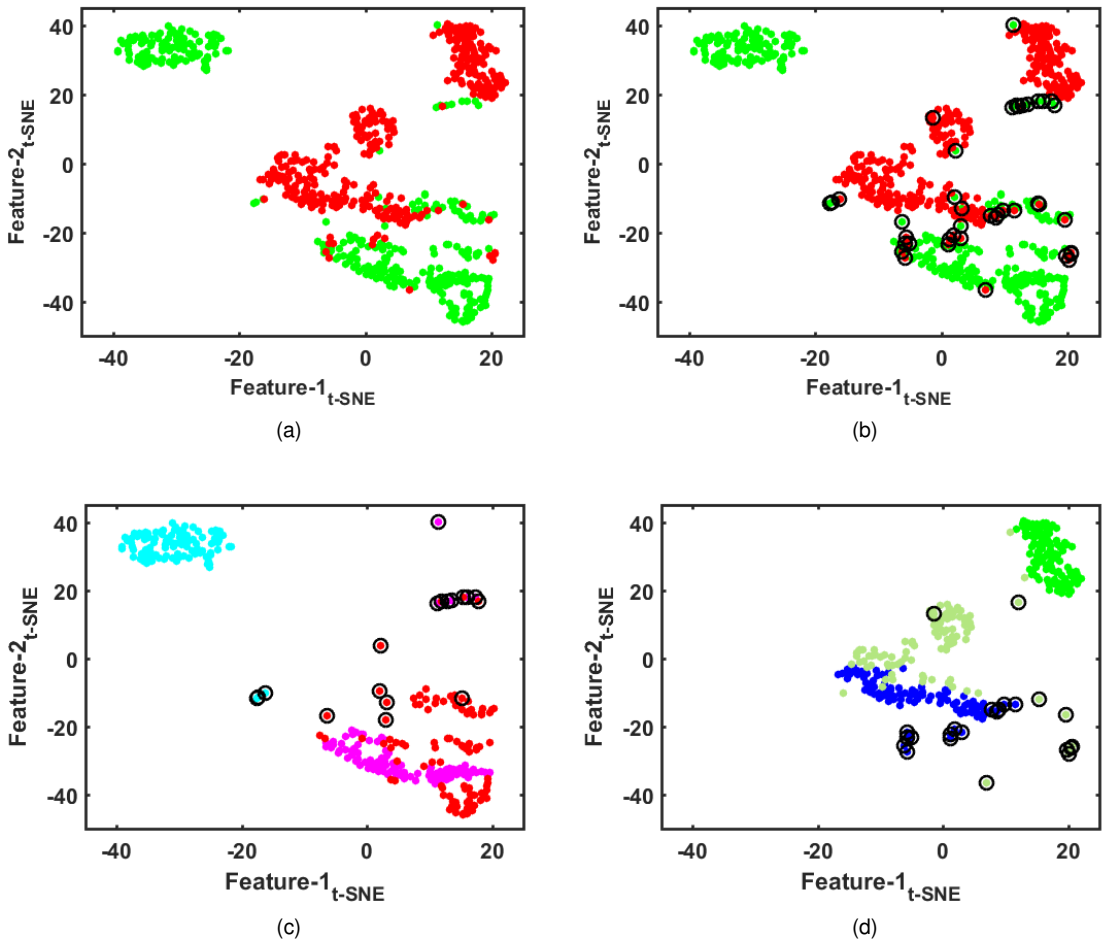
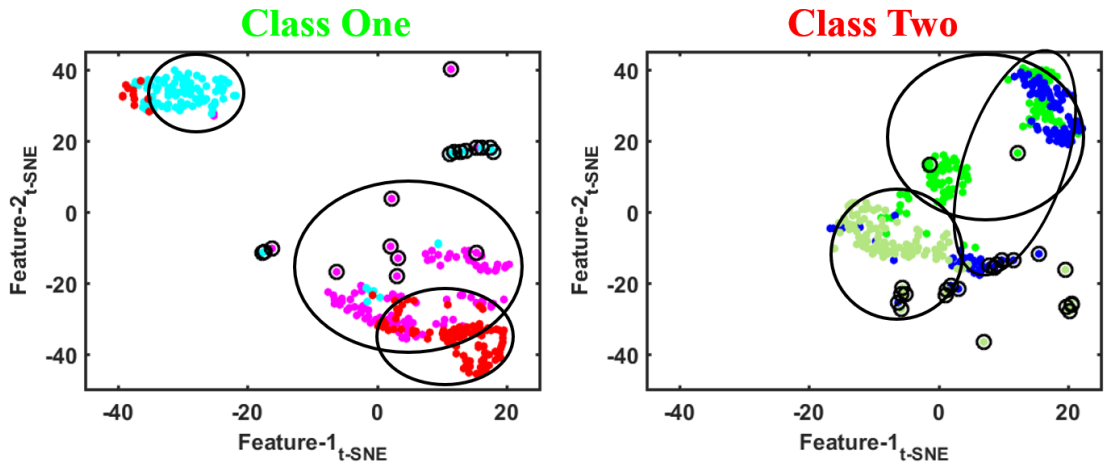
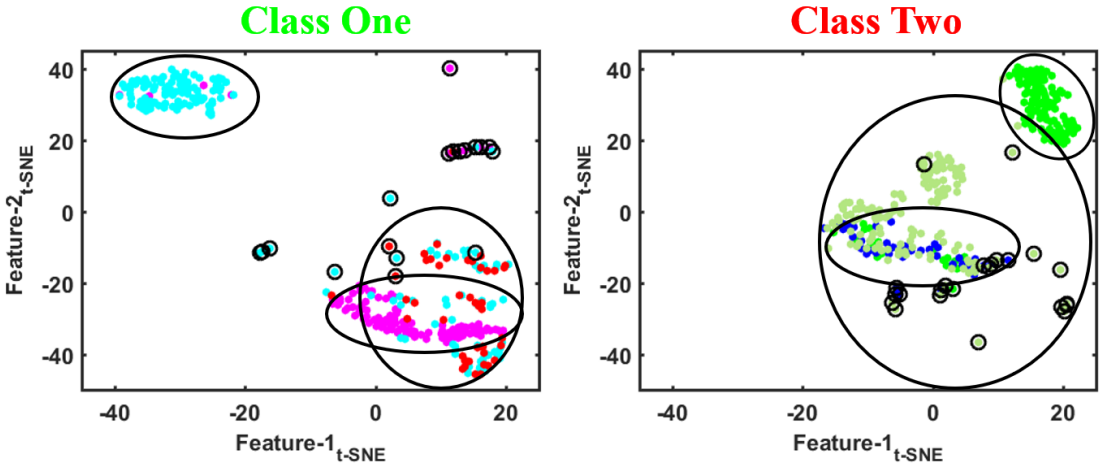


Figure 4.5: Sample distribution in the original space using the first two t-SNE dimensions. (a) Both classes, green for Class One and red for Class Two. (b) Both classes with boundary points marked by black circles. (c) Class One, where dots in three different colours represent samples generated by three different 5-variate normal distributions (d) Class Two, where dots in three different colours represent samples generated by three different 5-variate normal distributions.

The artificial data were created to contain two classes. Each class has 300 samples, in three subclasses with 100 samples per subclass. Samples in each subclass are generated by a 5-variate normal distribution, thus every sample is a vector of 5 feature values. So, this artificial data set is a 600 x 5 matrix, named as *artifi-600*. The sample distribution in the original space using two t-SNE dimensions is shown in Figure 4.5.

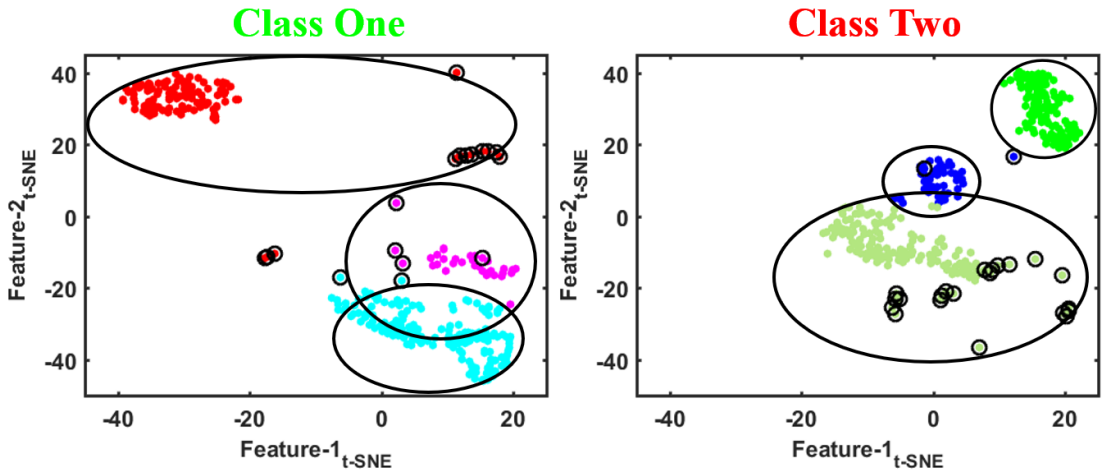


(a) SDA

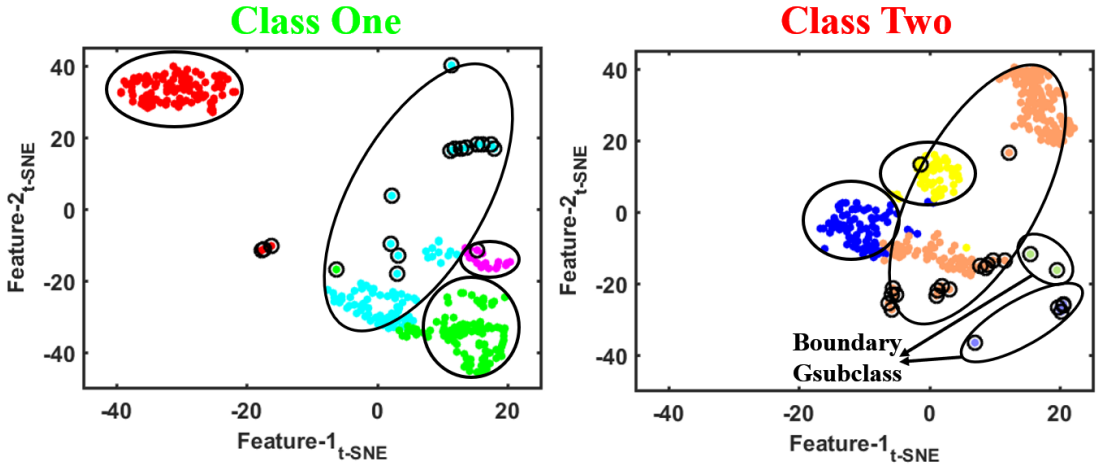


(b) MSDA

Figure 4.6: Sample distribution in the original space with subclasses found by SDA and MSDA methods. Subclasses are indicated by black circles, and represented by sample dots in different colours.



(a) SSDA



(b) GSDA

Figure 4.7: Sample distribution in the original space with subclasses found by SSDA and GSDA methods. Subclasses are indicated by black circles, and represented by sample dots in different colours.

4.3.1 Comparison in the Original Space

We compare the subclasses found by different methods as they are shown in the original data space. We use SDA, MSDA, SSDA, GSDA on *artifi-600* to find subclasses, which are shown in Figure 4.6 and Figure 4.7. According to Figure 4.5, Figure 4.6 and Figure 4.7, we have the following observations.

- Qualitatively, the subclasses found by GSDA are more clearly separable than those found by other methods. We visually inspect and compare the subclasses found by different methods. The subclasses found by SSDA and GSDA overlap much less than those found by SDA and MSDA. In particular, the subclasses found by GSDA are the least overlapping.

- Quantitatively, the separability of subclasses found by GSDA is higher than that of other methods, which is consistent with the visual perception from Figure 4.6 and Figure 4.7. We use the Dunn Index [30] to measure the degree of separability between subclasses. Dunn Index (DI) is commonly used to evaluate clustering algorithms. A higher DI indicates better clustering in that clusters are compact and well-separated from each other. The DIs of the subclasses in the original space found by SDA, MSDA, SSDA and GSDA are 0.0046, 0.0059, 0.0188 and 0.0222, respectively.

- Again, quantitatively, the known subclasses in the original space are better separated in GSDA space than in other space. The DIs of the six known subclasses in SDA, MSDA, SSDA and GSDA spaces are 0.0016, 0.0141, 0.0072 and 0.0144, respectively.

- GSDA can find natural clusters of data as subclasses, since it clusters (using the rough-refine algorithm) the whole data set rather than one class of data at a time. One example is the cluster of data in colour cyan at the top left corner of Figure

4.5(c). This cluster is part of one subclass generated by one normal distribution, and is well separated from other clusters in the original space. This cluster has been correctly identified as a single subclass by GSDA, which is represented by red dots in Figure 4.7(b), but not by any of the other methods.

- In terms of boundary data, GSDA can find boundary data and place them in separate subclasses, while this is not the case with SDA, MSDA and SSDA. For example, in Figure 4.7(b), two boundary subclasses are clearly marked, which contains Class Two samples that are mixed up with Class One samples, see Figure 4.5(b).

- In terms of the numbers of subclasses, GSDA finds more subclasses than SDA, MSDA and SSDA. For example, SDA/MSDA/SSDA found three subclasses for each of the classes, whereas GSDA found four subclasses and five subclasses for Class One and Class Two, respectively. (See Figure 4.6 and Figure 4.7). This provides evidence for the conclusion in Section 4.2.1 that the rough-refine algorithm of GSDA enables finding more subclasses for classes.

4.3.2 Comparison in the LDA Spaces

Now we compare the results of the different methods in different LDA spaces. These methods project data into respective LDA spaces spanned by a number of extracted LDA features. SDA found 4 LDA features, and MSDA, SSDA and GSDA all found 5 LDA features. In order to visualise data in the LDA space, we again use t-SNE to find two dimensions from each LDA space and plot data against these two dimensions (see Figure 4.8). Note that the classical LDA finds $C - 1$ LDA features for a data set with C classes, so the LDA space in Figure 4.8(b) has only one feature.

In Figure 4.8, the data samples are plotted in different spaces, with green and red representing the two classes. Comparing the original space with the classical LDA space, it is clear that the two classes are completely joined up in the classical LDA space, which

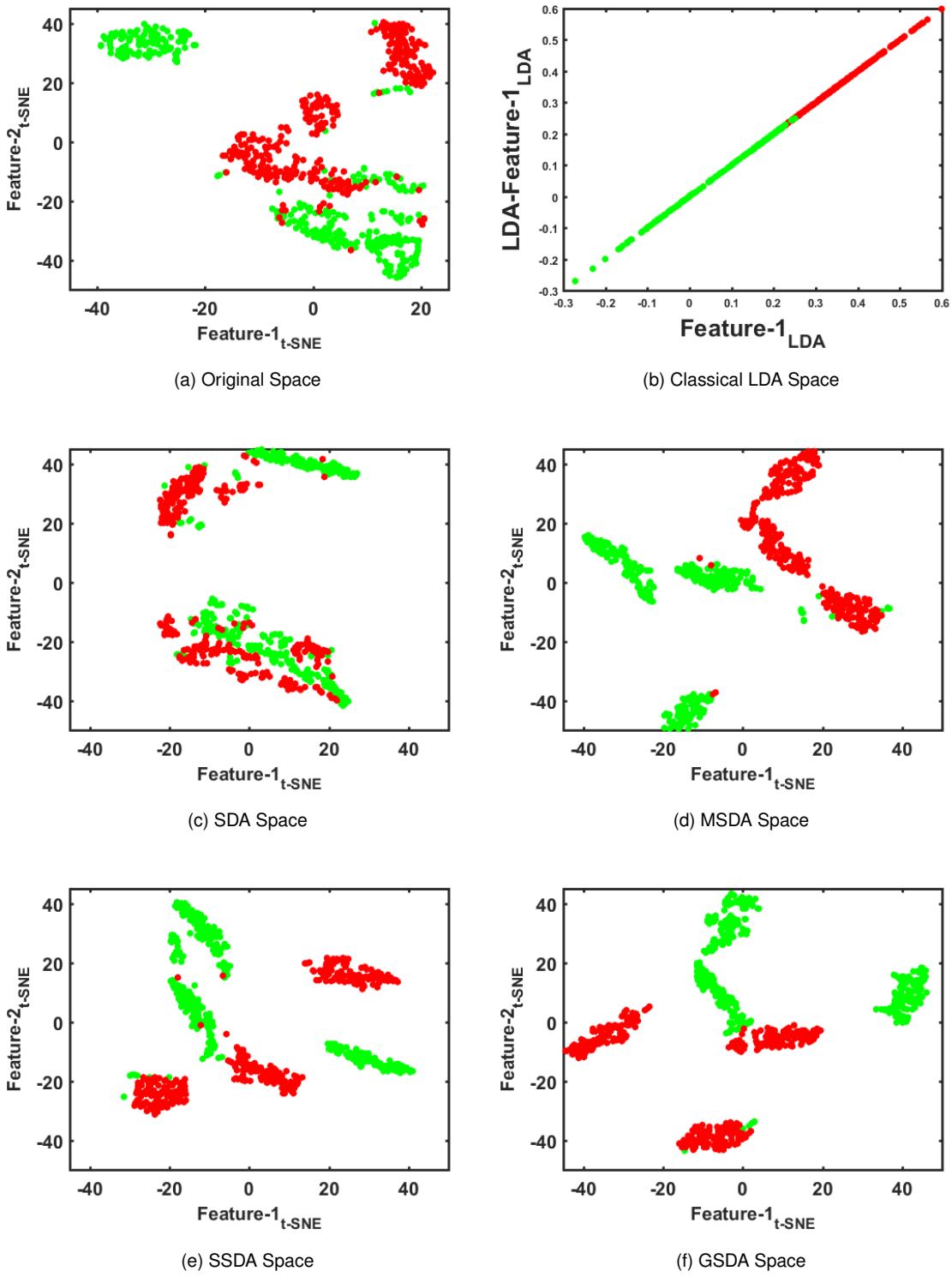
confirms the nonlinear data problem with the classical LDA. In the SDA space, the class separability does not improve much. However, the separability clearly improves substantially in MSDA, SSDA and GSDA spaces (see Figure 4.8(d-f)). Furthermore, three clusters in each class are clearly observable in these three spaces, which correspond to the three multivariate normal distributions in each class. In particular, six clusters in the GSDA space are more apparent than in the other spaces. Moreover, it is also clear that the two classes are better separated in the GSDA space than in MSDA and SSDA spaces: only a few green samples are mixed into the red class in GSDA, whereas some red and green samples are still mutually mixed in both the MSDA and SSDA spaces.

4.3.3 Summary

The comparative evaluations above support the following conclusions.

- GSDA can find subclasses that are less overlapping in the original data space than SDA/MSDA/SSDA.
- GSDA can find some subclasses at the class boundary, such as the two boundary subclasses shown in Figure 4.7(b).

We can then use the LDA optimisation process, coupled with the newly defined between-subclass and within-subclass scatter matrices, to more effectively separate different classes and also separate subclasses with every class.



(a) Original Space

(b) Classical LDA Space

(c) SDA Space

(d) MSDA Space

(e) SSDA Space

(f) GSDA Space

Figure 4.8: Sample distribution in different spaces, where different colours represent different classes.

4.4 Evaluations Using Real Data

In this section we use real data to evaluate the proposed GSDA through a series of experiments. Again we compare GSDA with four closely related linear discriminant analysis (DA) methods: LDA, SDA, MSDA and SSDA. We also compare GSDA with other nonlinear DA methods: Kernel LDA (KDA)[6], Kernel SDA (KSDA)[131] and Kernel MSDA (KMSDA)[39]. For these nonlinear methods, we employ five commonly used kernels: Gaussian radial basis (RBF) kernel, Gaussian kernel, Polynomial (Poly) kernel, PolyPlus kernel and Linear kernel. In our experiments, we consider a range of classification tasks: imbalanced classification, general classification and face recognition. Five data sets are selected from the KEEL [2] repository for imbalanced classification; ten data sets from the UCI Data Repository [28] for general classification; and YouTube faces database [125] for face recognition.

4.4.1 Data Sets and Notation

Five imbalanced data sets and ten UCI data sets are selected in the experiments. The data sets are all numerical, due to the need to compute the mean and distance. General information about these data sets is shown in Table 4.1 and Table 4.2, respectively.

Table 4.1: General information about the five imbalanced data sets used in experiments. In this table, #Class denotes the number of classes, #Attribute denotes the number of attributes, #Instance is the number of instances and IR means imbalanced ratio.

Name of dataset	#Class	#Attribute	#Instance	IR
Dermatology	6	34	366	5.55
Glass1	2	9	214	1.82
Hayes-roth(HR)	3	4	132	1.7
New-thyroid1(NT)	2	5	215	5.15
Wisconsin	2	9	683	1.86

We also use the YouTube faces database for the face recognition task. It contains 3,425 videos of 1,595 different people collected from YouTube. The average length of each video clip is 181.3 frames, and there are large variations in expression, pose and illumination in

Table 4.2: General information about ten UCI datasets used in experiments. Where FTM and WDBC are acronyms for forest type mapping and Wisconsin diagnostic breast cancer, respectively. #Class denotes the number of classes, #Attribute denotes the number of attribute and #Instance is the number of instances.

Name of dataset (Acronym)	#Class	# Attribute	#Instance
Diabetic	2	19	1,151
FTM	4	27	523
Glass	6	9	214
Haberman(HM)	2	3	306
Leaf	30	14	340
Pageblock(PB)	2	10	5,472
Penbased(Pen)	10	16	1,100
Pima	2	8	768
Seeds	3	7	210
WDBC	2	30	569

each video. In our experiments, we use the aligned images database, which contains aligned face frames broken from videos, and we use CenterSymmetric LBP (CSLBP) descriptor [47] provided by the YouTube faces database website to represent each face frame.

4.4.2 Experimental Results

In our experiments, every DA method mentioned above is applied to find a subspace where different classes are most separated based on the method's criteria. We project a data set to this subspace, use *k*-nearest neighbour (*k*NN, *k*=1) as the classifier and ten-fold cross-validation as the evaluation framework. The evaluation metrics are: *estimated mean accuracy* (EMA) and *standard error of the mean* (SEM): $EMA = \frac{\sum_{i=1}^{10} p_i}{10}$, where p_i denotes the percentage of correct classification in the ith fold validation; $SEM = \frac{\delta}{\sqrt{10}}$, where $\delta = \sqrt{\frac{\sum_{i=1}^{10}(p_i - EMA)^2}{9}}$. For the face recognition task, we use PCA to reduce data dimensionality and keep 95% of variance before the DA method is used.

4.4.2.1 Classification Accuracy: All Methods

Imbalanced Data: The classification accuracies of the linear and nonlinear DA methods on the five imbalanced data sets are shown in Table 4.3 and Table 4.4, respectively. We see from Table 4.3 that GSDA obtains the best classification accuracy on all imbalanced data

sets. In particular, GSDA outperforms LDA on *Glass1* by over 20%. Moreover, compared with the nonlinear DA methods, GSDA is the best on 4 out of 5 imbalanced data sets and second best on 1 (only 0.01% inferior to the best) according to Table 4.4.

Table 4.3: EMA±SEM of linear DA methods on five imbalanced data sets

Methods \ Datasets	Dermatology	Glass1	HR	NT	Wisconsin
LDA	0.9645±0.0134	0.6119±0.0226	0.7725±0.0499	0.9483±0.0150	0.9635±0.0090
GSDA	**0.9726±0.0072**	**0.8459±0.0197**	**0.8401±0.0437**	**1.0000±0.0000**	**0.9663±0.0038**
SSDA	0.9673±0.0068	0.7716±0.0318	0.8099±0.0478	0.9859±0.0072	0.9649±0.0082
SDA	0.9534±0.0092	0.7708±0.0317	0.7495±0.0374	0.9673±0.0142	0.9487±0.0059
MSDA	0.9679±0.0087	0.7721±0.0347	0.8099±0.0492	0.9952±0.0048	0.9634±0.0062

Table 4.4: EMA±SEM of GSDA and nonlinear DA methods on five imbalanced data sets

Methods \ Datasets	Dermatology	Glass1	HR	NT	Wisconsin
GSDA	0.9726±0.0072	**0.8459±0.0197**	**0.8401±0.0437**	**1.0000±0.0000**	**0.9663±0.0038**
KSDA(RBF)	0.9589± 0.0103	0.7987±0.0237	0.8176±0.0366	0.9907±0.0062	0.9619±0.0054
KMSDA(Gaussian)	0.3441±0.0285	0.6165±0.0330	0.6957±0.0244	0.5652±0.0022	0.4621±0.0440
KMSDA(Linear)	0.9589±0.0109	0.6972±0.0280	0.7495±0.0399	0.9812±0.0105	0.9605±0.0038
KMSDA(Poly)	0.9672±0.0037	0.7236±0.0284	0.8033±0.0323	0.9907±0.0062	0.9575±0.0051
KMSDA(PolyPlus)	**0.9727±0.0041**	0.7286±0.0317	0.8258±0.0318	0.9859±0.0072	0.9531±0.0084
KDA(Gaussian)	0.3135±0.0288	0.7236±0.0283	0.8104±0.0349	0.8556±0.0163	0.9194±0.0074
KDA(Linear)	0.9672±0.0068	0.7102±0.0182	0.7335±0.0491	0.9578±0.0111	0.9590±0.0081
KDA(Poly)	0.9482±0.0094	0.6732±0.0189	0.8176±0.0308	0.9859±0.0072	0.9517±0.0044
KDA(PolyPlus)	0.9453±0.0071	0.6405±0.0293	0.8110±0.0361	0.9905±0.0063	0.9663±0.0085

UCI Data: Results on the ten UCI data sets are shown in Table 4.5 and Table 4.6. It can be observed from Table 4.5 that GSDA achieves better classification accuracy than SDA and MSDA on the majority of data sets. Furthermore, GSDA outperforms LDA on all ten data sets. In addition, compared with SSDA, GSDA appears to be on a par with it on these UCI data sets. As can be seen from Table 4.5, GSDA and SSDA were close in classification performance on all of the UCI data sets. In particular, for HM, and Pen data sets, the difference between GSDA and SSDA is less than 0.001. Additionally, comparing GSDA with kernel DA methods, it is noted that GSDA is superior than KSDA and different types of KMSDA on 6 out of 10 and 8 out of 10 UCI data sets, respectively. Compared with different types of KDA, GSDA outperforms linear or poly KDA and polyplus KDA

Table 4.5: EMA±SEM of linear DA methods on ten UCI data sets

Methods \ Datasets	Diabetic	FTM	Glass	HM	Leaf	PB	Pen	Pima	Seeds	WDBC
LDA	0.6377 ± 0.0128	0.8468 ± 0.0142	0.5747 ± 0.0269	0.6603 ± 0.0196	0.6912 ± 0.0417	0.9117 ± 0.0026	0.9582 ± 0.0043	0.6782 ± 0.0216	0.9524 ± 0.0142	0.9579 ± 0.0079
GSDA	0.6699 ± 0.0131	**0.8871** ± **0.0106**	0.6775 ± 0.0303	0.7025 ± 0.0216	**0.7765** ± **0.0197**	0.9439 ± 0.0030	0.9800 ± 0.0052	0.6939 ± 0.0216	**0.9619** ± **0.0119**	0.9613 ± 0.0090
SSDA	**0.6907** ± **0.0103**	0.8603 ± 0.0175	0.6647 ± 0.0372	**0.7029** ± **0.0229**	0.7676 ± 0.0242	0.9673 ± 0.0022	**0.9809** ± **0.0039**	**0.7004** ± **0.0157**	0.9524 ± 0.0142	0.9632 ± 0.0088
SDA	0.6508 ± 0.0144	0.8641 ± 0.0121	**0.7106** ± **0.0273**	0.6962 ± 0.0307	0.5971 ± 0.0186	0.9311 ± 0.0034	0.9736 ± 0.0055	0.6913 ± 0.0148	**0.9619** ± **0.0119**	0.9226 ± 0.0088
MSDA	0.6846 ± 0.0080	0.8737 ± 0.0175	0.6262 ± 0.0291	0.6898 ± 0.0295	0.7265 ± 0.0260	**0.9678** ± **0.0022**	0.9791 ± 0.0041	0.6965 ± 0.0150	0.9571 ± 0.0085	**0.9648** ± **0.0045**

Table 4.6: EMA±SEM of GSDA and nonlinear DA methods on ten UCI data sets

Methods \ Datasets	Diabetic	FTM	Glass	HM	Leaf	PB	Pen	Pima	Seeds	WDBC
GSDA	0.6699 ± 0.0131	**0.8871** ± **0.0106**	0.6775 ± 0.0303	0.7025 ± 0.0216	0.7765 ± 0.0197	0.9439 ± 0.0030	0.9800 ± 0.0052	0.6939 ± 0.0216	0.9619 ± 0.0119	**0.9613** ± **0.0090**
KSDA(RBF)	**0.6898** ± **0.0150**	0.8870 ± 0.0127	0.6491 ± 0.0114	0.6861 ± 0.0164	0.7588 ± 0.0209	**0.9627** ± **0.0015**	**0.9827** ± **0.0050**	0.7055 ± 0.0175	0.9381 ± 0.0201	0.9402 ± 0.0076
KMSDA(Gaussian)	0.5308 ± 0.0168	0.3725 ± 0.0188	0.6165 ± 0.0330	**0.7608** ± **0.0284**	**0.8187** ± **0.0362**	0.8883 ± 0.0186	0.9161 ± 0.0030	0.0764 ± 0.0049	0.6509 ± 0.0216	0.2435 ± 0.0144
KMSDA(Linear)	0.6751 ± 0.0096	0.8603 ± 0.0163	**0.7063** ± **0.0229**	0.6931 ± 0.0280	0.7559 ± 0.0224	0.9611 ± 0.0020	0.9782 ± 0.0041	0.6770 ± 0.0117	0.9333 ± 0.0105	0.9350 ± 0.0094
KMSDA(Poly)	0.6560 ± 0.0077	0.8717 ± 0.0141	0.6721 ± 0.0250	0.6700 ± 0.0279	0.7559 ± 0.0248	0.9560 ± 0.0027	0.9791 ± 0.0038	0.6847 ± 0.0165	0.9667 ± 0.0160	0.9403 ± 0.0126
KMSDA(PolyPlus)	0.6802 ± 0.0143	0.8659 ± 0.0147	0.6677 ± 0.0248	0.6959 ± 0.0308	0.7529 ± 0.0253	0.9552 ± 0.0027	0.9782 ± 0.0045	0.6848 ± 0.0139	0.9714 ± 0.0127	0.9279 ± 0.0106
KDA(Gaussian)	0.4856 ± 0.0172	0.3287 ± 0.0227	0.6348 ± 0.0194	0.5917 ± 0.0192	0.7559 ± 0.0201	0.9086 ± 0.0034	0.0991 ± 0.0080	0.5937 ± 0.0449	0.9190 ± 0.0225	0.6081 ± 0.0282
KDA(Linear)	0.6404 ± 0.0109	0.8467 ± 0.0168	0.6764 ± 0.0249	0.6280 ± 0.0294	0.7706 ± 0.0227	0.9327 ± 0.0031	0.9564 ± 0.0073	0.6978 ± 0.0155	0.9571 ± 0.0132	0.9315 ± 0.0112
KDA(Poly)	0.6603 ± 0.0129	0.8755 ± 0.0142	0.6459 ± 0.0244	0.6573 ± 0.0312	0.7088 ± 0.0188	0.8955 ± 0.0055	0.9782 ± 0.0043	0.6768 ± 0.0189	0.9667 ± 0.0102	0.9262 ± 0.0097
KDA(PolyPlus)	0.6559 ± 0.0129	0.8717 ± 0.0116	0.6922 ± 0.0213	0.6116 ± 0.0325	0.7088 ± 0.0208	0.8896 ± 0.0083	0.9755 ± 0.0054	**0.7067** ± **0.0266**	**0.9714** ± **0.0105**	0.9121 ± 0.0091

on 9 out of 10 and 7 out of 10 data sets, respectively. Furthermore, GSDA achieves better classification accuracy than Gaussian KDA on all UCI data sets.

Face data: Results for linear DA and nonlinear DA on YouTube are presented in Table 4.7 and Table 4.8, respectively. It can be readily seen that GSDA is superior to LDA, SDA, MSDA and SSDA on YouTube data set. Compared with the nonlinear DA methods, GSDA is also quite competitive. From Table 4.8 we can observe that GSDA obtains better classification accuracy than the majority of the nonlinear DA methods on YouTube data set.

Table 4.7: EMA±SEM of linear DA methods on YouTube

Methods \ Datasets	YouTube
LDA	0.9790 ± 0.0043
GSDA	$\mathbf{0.9820 \pm 0.0035}$
SSDA	0.9790 ± 0.0050
SDA	0.9760 ± 0.0050
MSDA	0.9790 ± 0.0043

Table 4.8: EMA±SEM of GSDA and the nonlinear DA methods on YouTube

Methods \ Datasets	YouTube
GSDA	0.9820 ± 0.0035
KSDA(RBF)	0.9750 ± 0.0050
KMSDA(Gaussian)	0.9850 ± 0.0040
KMSDA(Linear)	0.9780 ± 0.0039
KMSDA(Poly)	0.9810 ± 0.0043
KMSDA(Polyplus)	0.9840 ± 0.0034
KDA(Gaussian)	0.9740 ± 0.0033
KDA(Linear)	0.9780 ± 0.0049
KDA(Poly)	0.9790 ± 0.0041
KDA(PolyPlus)	$\mathbf{0.9850 \pm 0.0034}$

4.4.2.2 Runtime Performance

Runtime results for all DA methods used in our experiments are shown in Table 4.9, 4.10, 4.11 and 4.12. It is not surprising that GSDA is slower than LDA on all data sets. However, GSDA is faster than SDA, MSDA, SSDA and all nonlinear DA methods on most of the data sets. In particular, GSDA is faster than KSDA and all KMSDAs on all data sets. Furthermore, GSDA is much faster than all nonlinear DA methods on data sets that

have large numbers of samples, such as *Diabetic*, *Pageblock*, *Penbased* and *YouTube*. This is because constructing the Gram matrix, needed in these nonlinear DA methods, is time consuming with time complexity of $O(N^2)$, where N is the number of samples.

Table 4.9: Running time,in seconds, of the DA methods on five imbalanced data sets

Methods \ Datasets	Dermatology	Glass1	HR	NT	Wisconsin
LDA	1.6557	0.2300	0.1300	0.1229	0.1543
SDA	2.7748	1.1478	0.2902	0.3631	0.9264
MSDA	52.4797	5.2600	0.9648	0.3930	0.6126
SSDA	2.2682	0.3298	0.6505	0.2223	0.3368
GSDA	2.1623	0.7536	0.4642	0.2219	0.6060
KSDA(RBF)	19.0836	7.4625	15.6092	4.7507	594.8892
KMSDA(Gaussian)	141.0202	20.3256	13.2208	4.0478	150.7365
KMSDA(Linear)	131.7647	17.7479	31.5338	7.5950	123.1154
KMSDA(Poly)	66.0409	24.4430	31.4333	9.7744	75.2237
KMSDA(PolyPlus)	86.8109	26.2986	7.8135	13.1022	165.4497
KDA(Gaussian)	5.5623	1.1765	0.3752	1.6523	20.2635
KDA(Linear)	3.3656	0.3266	0.2306	0.3068	7.0781
KDA(Poly)	3.5787	0.3447	0.3278	0.7560	7.2276
KDA(PolyPlus)	3.5375	0.3404	0.3437	0.7405	7.3179

Table 4.10: Running time,in seconds, of the DA methods on the first five UCI data sets

Methods \ Datasets	Diabetic	FTM	Glass	HM	Leaf
LDA	0.1900	0.1650	0.1424	0.1233	0.3886
SDA	2.6287	0.8241	0.3431	0.4253	1.5963
MSDA	3.3307	8.1953	2.6736	1.5004	212.4678
SSDA	3.0190	0.6298	0.4516	0.4313	1.7393
GSDA	1.7140	0.3242	0.7146	0.2269	0.4141
KSDA(RBF)	213.2813	23.6540	6.0525	3.6151	42.7697
KMSDA(Gaussian)	349.3192	10.5250	7.0517	37.7344	222.6590
KMSDA(Linear)	297.0769	102.6789	12.4334	31.2964	213.6003
KMSDA(Poly)	355.0283	51.244	12.9038	15.4441	219.3206
KMSDA(PolyPlus)	228.5800	106.9702	6.3781	19.3184	215.7691
KDA(Gaussian)	169.3253	7.2256	1.3528	2.9344	2.4062
KDA(Linear)	41.3916	1.2954	0.4064	0.4665	1.9036
KDA(Poly)	48.3372	4.9327	0.3643	0.4720	2.2376
KDA(PolyPlus)	52.6573	6.5416	0.3680	0.4465	2.0935

Table 4.11: Running time,in seconds, of the DA methods on the rest of UCI data sets

Methods \ Datasets	PB	Pen	Pima	Seeds	WDBC
LDA	0.4218	0.1937	0.1383	0.1140	0.1590
SDA	27.0573	1.3301	2.2941	0.3595	1.3104
MSDA	11.7562	3.7752	2.3198	0.2801	0.9236
SSDA	23.9614	1.0710	1.0191	0.2708	0.8599
GSDA	29.9126	3.1126	0.8809	0.2380	1.2136
KSDA(RBF)	8093.8562	209.9351	46.0343	1.3690	181.9985
KMSDA(Gaussian)	60107.4813	410.1540	249.0932	86.4801	149.7931
KMSDA(Linear)	23502.7624	362.4776	205.7105	26.2279	107.4712
KMSDA(Poly)	30886.4855	399.8886	113.1669	31.4010	82.2190
KMSDA(PolyPlus)	38842.8943	136.9045	169.2872	3.5144	70.4129
KDA(Gaussian)	13601.9607	53.0902	48.3499	1.7390	14.8941
KDA(Linear)	2979.7338	8.0055	13.7801	0.2875	0.5328
KDA(Poly)	3859.043	35.6853	12.2930	0.3103	7.3115
KDA(PolyPlus)	3444.0761	35.6975	13.2514	0.3010	6.3650

Table 4.12: Running time in seconds, of the DA methods on YouTube

Methods \ Datasets	YouTube
LDA	4.7006
SDA	12.0743
MSDA	31730.1390
SSDA	7.0082
GSDA	10.7876
KSDA(RBF)	8207.4686
KMSDA(Gaussian)	116396.1573
KMSDA(Linear)	121923.7487
KMSDA(Poly)	169384.5576
KMSDA(PolyPlus)	116007.9136
KDA(Gaussian)	81.5227
KDA(Linear)	42.0474
KDA(Poly)	78.0980
KDA(PolyPlus)	77.8868

4.5 Summary

This chapter has presented a new subclass-based variant of LDA, *global subclass discriminant analysis (GSDA)*, to deal with the nonlinear data problem when different classes can not be linearly separated in the original data space. The new method is designed to address a problem with existing subclass-based LDA variants when subclasses are selected locally, i.e. based on data only from individual classes, thus missing opportunities. GSDA finds subclasses for each class by applying a rough-refine clustering algorithm to the whole data set, rather than one class of data at a time. These subclasses are thus called global subclasses or G-subclasses. Then GSDA finds a subspace that maximises the average distance between these G-subclasses and concurrently minimises the average distance within every G-subclass, where data become linearly separable at the subclass level. This is achieved by re-defining the Fisher-Rao's criterion using new scatter matrices, one for between G-subclasses and one for within G-subclasses, and then applying the LDA optimisation process.

Extensive experiments using a variety of challenging data sets and a mixture of linear discriminant analysis methods and non-linear ones have produced convincing results to conclude that GSDA is a new state-of-the-art method for discriminant analysis. GSDA has outperformed linear DA methods consistently, and also outperformed non-linear DA methods in most of our experiments in terms of both accuracy and runtime. In particular, GSDA has outperformed both linear and non-linear DA methods on imbalanced data sets. This suggests that GSDA is a competitive solution to the challenging problem of imbalanced classification.

We have argued, using examples, that this superior performance is due to the fact that GSDA is able to find some clusters of class boundary data, and then uses the powerful LDA mechanism to push different classes apart more effectively, which is supported by our experiments in Section 4.3.

CHAPTER V

Cluster-based Data Relabelling for Classifier

In Chapter IV it was found that, by finding global clusters including those that contain multiple classes (i.e. boundary clusters) and then extracting features that separate classes based on these global clusters, significant classification performance gains have been achieved. In this chapter, we generalise this cluster-based idea to benefit all classifiers especially linear classifiers, in a new method called *cluster-based data relabelling* (CBDR).

5.1 Motivation

The classifier is a key process in supervised classification that aims to learn classification functions/decision boundaries to classify samples into different classes. It is used in many different applications such as text classification [60, 89], computer vision [121, 129] and pattern recognition [132, 94]. There are many classifiers in the literature, broadly grouped into linear classifiers and nonlinear classifiers. Linear classifiers are a class of classifiers that use linear classification functions to classify data. Well-known linear classifiers include linear discriminant analysis classifier (LDAC) [33, 85], linear support vector machine (LSVM) [14] and linear multilayer perceptron (LMLP) [110].

Linear classifiers are simple to train and use, and work well on linearly separable data, or *linear data* for short. However, they tend to have unsatisfactory classification

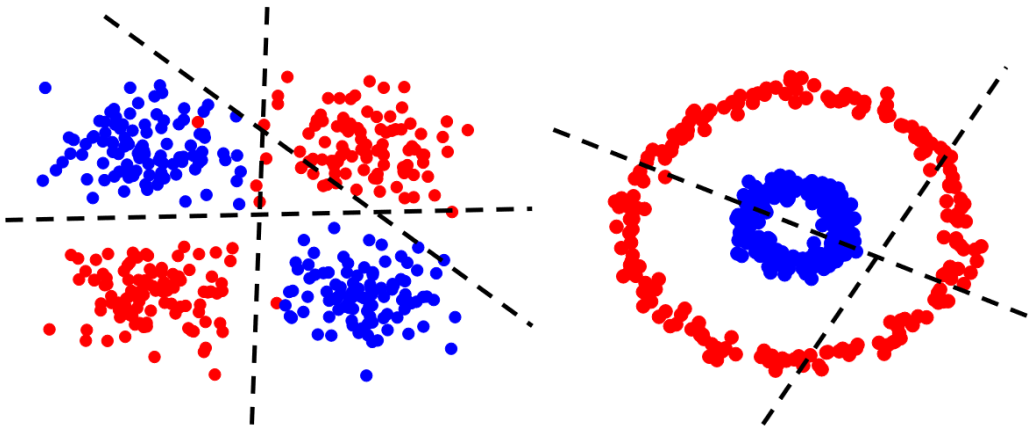


Figure 5.1: Examples of nonlinear data, where different colour dots represent samples from different classes, dashed black lines represent linear decision boundaries.

performance on linearly inseparable data, or *nonlinear data* for short, since linear decision boundaries have limited discriminative power. Consider, for example, the nonlinear data sets in Figure 5.1. Clearly it is difficult to find a linear decision boundary to separate the two classes in each data set. We call this the *nonlinear problem* of linear classifiers. There are many real data sets that are linearly inseparable, therefore, linear classifiers have restricted applications.

Linear classifiers can be extended to work on nonlinear data, resulting in their nonlinear variants. LSVM has been extended using a technique called the *kernel trick*. The idea of the kernel trick is to use Mercer kernel functions such as radial basis kernel function (RBF) and polynomial kernel function (Poly) [10], to implicitly map nonlinear data to a high-dimensional space $\mathcal{F}$, where data become linearly separable. Then LSVM is able to separate the data well in the space $\mathcal{F}$. The decision boundaries found by LSVM based on RBF and Poly kernel functions are linear in the space $\mathcal{F}$, but nonlinear in the original space, so the extended LSVM is a nonlinear classifier, denoted by NLSVM for short. Although NLSVM outperforms LSVM on nonlinear data, it comes with an additional hyper parameter, the kernel function. There is no kernel function that works well for all data sets, so it is necessary to identify a suitable kernel function for a specific data set before NLSVM is applied. There is no general guideline for this, so trial and error

is usually needed. Unlike nonlinear SVM, quadratic discriminant analysis (QDA) [44] extends LDAC to work on nonlinear data by relaxing the assumption of LDAC, that is, that each class has a normal distribution with the same covariance matrix. QDA allows every class to have a different covariance matrix, thus resulting in a nonlinear discriminant analysis classifier. A common method to extend LMLP as a nonlinear classifier is to use a nonlinear activation function (e.g. sigmoid or hyperbolic tangent) to replace the linear activation function used in LMLP. Apart from linear to nonlinear extensions as explained above, there are other extensions of the classifiers such as piecewise-linear discriminant analysis [40], quasi-linear SVM [136], mixing linear SVM [36], and locally linear SVM [127].

Although there are many methods for extending linear classifiers to deal with nonlinear data or other complex data, most of them are specific to a particular linear classifier and so are not applicable to other classifiers. It is therefore desirable to have a method that can be applied to extend any linear classifier and simultaneously enhance their classification performance on nonlinear data. In this chapter, we propose to extend linear classifiers to work on nonlinear data by restructuring the data instead of modifying the classifier. Motivated by *separability-oriented subclass discriminant analysis* (SSDA) [119] and *global subclass discriminant analysis* (GSDA), we further propose to restructure the data by clustering data into class-specific clusters, taking the class-specific clusters as separate classes, and relabelling data by the cluster they belong to. We call this method *cluster-based data relabelling* (CBDR). More specifically, CBDR partitions the whole data set into several non-overlapping class-specific clusters, so that data samples in different clusters are linearly separable. Each cluster is taken as a separate class, and data samples are relabelled by the cluster they belong to. Since clustering is done on a per-class basis, there is a simple mapping between the original class labels and the new labels. After the data set is relabelled in this way, a linear classifier, in fact any classifier, can be applied to build a classification model. The class labels generated by this model are

new labels which must be converted into the original labels. It is worth noting that the classification model built from a relabelled data set has cluster-based decision boundaries instead of class-based ones.

The proposed data relabelling method has some advantages:

- A linear classifier equipped with CBDR preserves all its original benefits, including efficiency and interpretability but can effectively handle nonlinear data.
- CBDR is a generic method for extending linear classifiers.
- CBDR can also be used with nonlinear classifiers.

We conducted extensive experimentation using CBDR together with three classic linear classifiers, LDAC, LSVM and LMLP, and two nonlinear classifiers, decision trees and naive Bayes, on a wide range of real data. Significant improvements have been demonstrated, showing the utility of the proposed CBDR-based classification.

The rest of this chapter is organised as follows. Section 5.2 describes the details of the CBDR-based classification method. Section 5.3 presents the experimental results. Section 5.4 concludes the chapter.

5.2 Cluster-based Data Relabelling

Linear classifiers are extended to work on nonlinear data usually by modifying the algorithms, resulting in their nonlinear variants. These extension methods tend to be specific to the classification algorithms, so they may not be applicable to extend other linear classifiers. We propose to enable linear classifiers to work on nonlinear data by restructuring data in such a way that the restructured data can be linearly separated. Since we do not modify the classifier, this extension method works for any linear classifier.

In this section we present a data restructuring method, *cluster-based data relabelling* (CBDR), which is motivated by SSDA [119] and GSDA. CBDR applies a clustering

algorithm to the whole data set, dividing the data set into several non-overlapping class-specific clusters. It then relabels data by the clusters they belong to. Since clusters are class-specific, there is a simple mapping between new class labels and original ones. The CBDR method is described in detail in Algorithm 3.

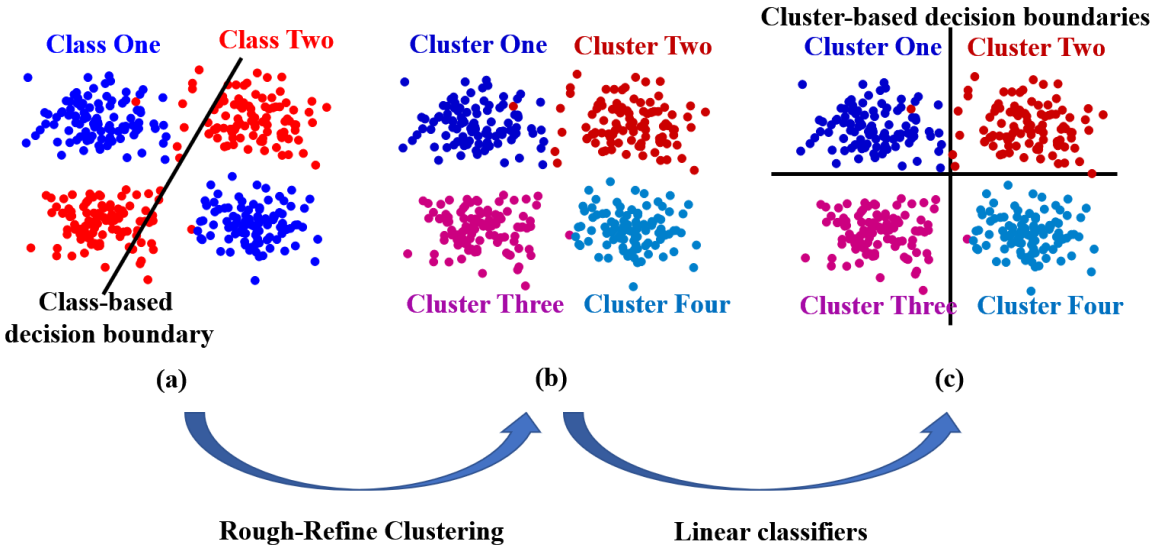


Figure 5.2: Illustration of CBDR, which enables separation of nonlinear data in a linear way, through clustering and relabelling. (a) shows samples of training data from the class perspective, where there are two classes, blue dots represent samples from Class One, red dots represent samples from Class Two, and the black straight line represents a class-based decision boundary. (b) shows samples of training data from the cluster perspective, where there are four clusters; different coloured dots represent samples from different clusters. (c) shows the cluster-based decision boundaries found by linear classifiers equipped with CBDR, where the black straight lines represent the cluster-based decision boundaries.

To justify CBDR, consider the example in Figure 5.2. It is clear that Class One and Class Two in Figure 5.2(a) can't be separated by a straight line. However, there are clearly

Algorithm 3 Cluster-based data relabelling (CBDR).

Input: T_{set} and k_{max}. $T_{set} = \{s_{ij}|i = 1, 2, ..., n; j = 1, 2, ..., n_i\}$ is the training set, where s_{ij} represents the jth sample of the ith class, n is the number of classes and n_i is the number of samples in the ith class. k_{max} is the maximum number of clusters, which is a parameter of the rough-refine clustering algorithm (RRC).

Output: L_{class}. $L_{class} = \{l_{ij}|i = 1, 2, ..., n, j = 1, 2, ..., n_i\}$ is the predicted class label set, where l_{ij} represents the class label of the jth sample in the ith class.

1: Apply RRC algorithm on T_{set} with k_{max} to obtain global cluster labels – $GCluLabel = \{a_{ij}|i = 1, 2, .., n; j = 1, 2, ..., n_i\}$, and the distribution of cluster – $Dclu = \{M_i|i = 1, 2, .., n\}$, where a_{ij} represents the cluster label of the jth sample in the ith class, $M_i = \{l_{ik}|i = 1, 2, .., n; k = 1, 2, ..., K_i\}$ consists of the cluster labels belonging to the ith class, l_{ik} is the cluster label of the kth cluster in the ith class, K_i is the number of clusters in the ith class.
2: $GCluLabel$, $Dclu$ = RRC(T_{set}, k_{max});
3: Relabel T_{set} by using $GCluLabel$ as new class labels, then apply the classifier to find the cluster-based decision boundaries $\mathcal{D}s$.
4: $\mathcal{D}s = Classifier(T_{set}, GCluLabel)$;
5: Utilise the found $\mathcal{D}s$ to obtain the predicted cluster labels of T_{set}. $L_{cluster} = \{b_{ij}|i = 1, 2, .., n; j = 1, 2, ..., n_i\}$ is the set of the predicted cluster labels, where b_{ij} represents the predicted cluster label of the jth sample in the ith class by $\mathcal{D}s$.
6: $L_{cluster} = \mathcal{D}s(T_{set})$;
7: $L_{class} = \{\}$;
8: **for** $i = 1$ to n **do**
9: **for** $j = 1$ to n_i **do**
10: **if** cluster label b_{ij} belongs to M_i **then**
11: $l_{ij} = i$;
12: $L_{class} = L_{class} \cup \{l_{ij}\}$;
13: **else**
14: $l_{ij} = null$;
15: $L_{class} = L_{class} \cup \{l_{ij}\}$;
16: **end if**
17: **end for**
18: **end for**
19: **return** L_{class};

four clusters (see Figure 5.2(b)) in this data set, and they can be separated by a set of straight lines or, in other words, they are linearly separable. Therefore, although the two classes are not linearly separable, the four clusters are. If the mapping between the clusters and the classes is established, we can build a linear classifier to classify data by cluster labels and, based on the cluster mapping, to further classify by class labels. Figure 5.2(c) shows the cluster-based decision boundaries, which perfectly separate Class One and Class Two in an indirect way.

Clustering is a key operation in CBDR. A straightforward approach is to cluster every class into several clusters in the same way as in SSDA [119]. This approach considers only the data distribution of a class, without considering any other classes, thus losing the class separating information at class boundaries that is important for accurate classification. What matters most is the data distribution of the whole data set. This leads to the *rough-refine clustering* approach proposed in GSDA, which is adopted in this chapter. Instead of clustering every class, we cluster the whole data set into several clusters and then create class-specific clusters. More specifically, the clustering process is divided into two parts, namely *rough clustering* and *refine clustering*. The rough clustering clusters the whole data set irrespective of their class memberships, resulting in *global clusters*. These global clusters carry the important class separating information at the class boundaries. The refine clustering obtains class-specific clusters from the *global clusters*, and these class-specific clusters carry the structure information of each class. The *rough-refine clustering* algorithm is presented in detail in Chapter IV.

5.3 Experiments

In this section, we evaluate the performance of CBDR using a variety of artificial and real-world data. Without loss of generality, we use CBDR along with one of three classic linear classifiers, LDAC, LSVM and LMLP, resulting in three "extended" classifiers, LDAC-CBDR, LSVM-CBDR and LMLP-CBDR. Generalisation to other linear classifiers

is straightforward. The linear classifiers used in the experiments can be found in MATLAB R2016b [84].

5.3.1 Evaluation Using Artificial Data

To evaluate the utility of CBDR and visually illustrate how CBDR works, we first use three typical nonlinear data sets: *xor*, *moon* and *circle*. In this section, we evaluate CBDR from two perspectives: 1) we compare the linear classifiers with their CBDR-based extensions on decision boundaries and classification accuracies, including LDAC *vs* LDAC-CBDR, LSVM *vs* LSVM-CBDR and LMLP *vs* LMLP-CBDR; 2) the linear classifiers equipped with CBDR are compared against common nonlinear classifiers, including LDAC-CBDR *vs* QDA, LSVM-CBDR *vs* RSVM and PSVM, and LMLP-CBDR *vs* TSMLP, where RSVM, PSVM and TSMLP denote SVM with RBF kernel, SVM with Poly kernel of degree two, and MLP with hyperbolic tangent sigmoid function, respectively.

There are two classes in each of *xor*, *moon* and *circle*. Every class has 200 samples and each sample has two features. So, these three nonlinear data sets are 400 x 2 matrices. The sample distributions of the three data sets are shown in Figure 5.3, where different colours represent different classes. We can note from Figure 5.3 that it is hard to separate the two classes by using one straight line. In our experiments, each data set is split into the training data set and the testing data set. There are 200 samples in the training data set, where 100 samples are randomly chosen from each class. The remaining data is taken as the testing data.

Comparisons between linear classifiers and their CBDR-equipped extensions: From Figure 5.4(a), Figure 5.5(a) and Figure 5.6(a), it does not surprise us that the decision boundaries found by LDAC, LSVM and LMLP for the three nonlinear data sets are all straight lines, and that they all fail to separate the classes. There are blue dots and red dots misplaced on each side of decision boundaries. In particular, for the *xor* data set, all three

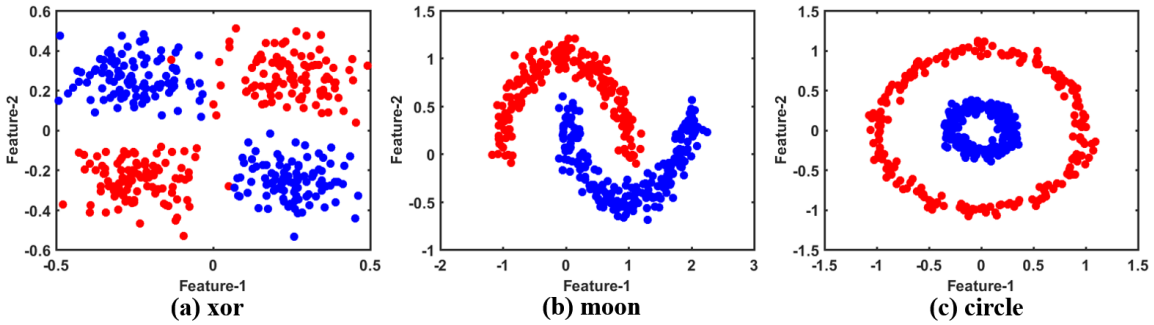
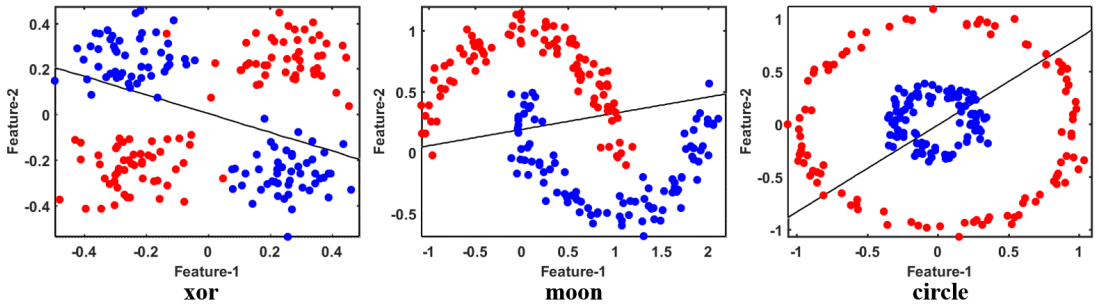
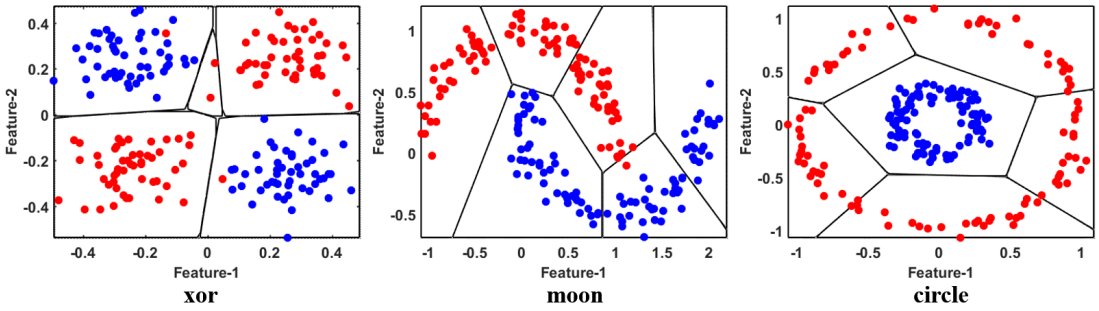


Figure 5.3: Sample distributions of three typical nonlinear data sets.

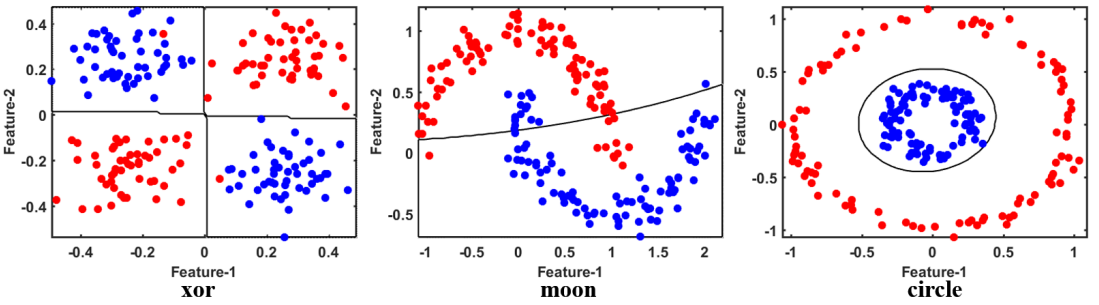
linear classifiers perform poorly, since one half of the training data is misclassified. By contrast, the decision boundaries found by LDAC-CBDR, LSVM-CBDR and LMLP-CBDR are much better. We can see clearly from Figure 5.4(b), Figure 5.5(b) and Figure 5.6(b) that red dots and blue dots are separately distributed on the each side of every decision boundary, which means that samples from each class are well separated. So, qualitatively, we can conclude that decision boundaries found by linear classifiers equipped with CBDR are superior to the ones found by the linear classifiers without CBDR. Furthermore, we evaluate these decision boundaries on the testing data, and compare them in terms of classification accuracy. The classification accuracies of all classifiers on the three artificial data sets are shown in Figure 5.7, Figure 5.8 and Figure 5.9, respectively. It is clearly seen from the bar charts that linear classifiers equipped with CBDR significantly outperform the linear classifiers without CBDR on all three data sets. LDAC-CBDR, LMLP-CBDR and LSVM-CBDR surpass their corresponding linear classifiers by over 40% on the *xor* data set, and LDAC-CBDR even surpasses LDAC by over 54% on the circle data set. Both qualitative and quantitative analyses consistently demonstrate that CBDR can enhance the classification performance of linear classifier on nonlinear data.



(a) Decision boundaries found by LDAC

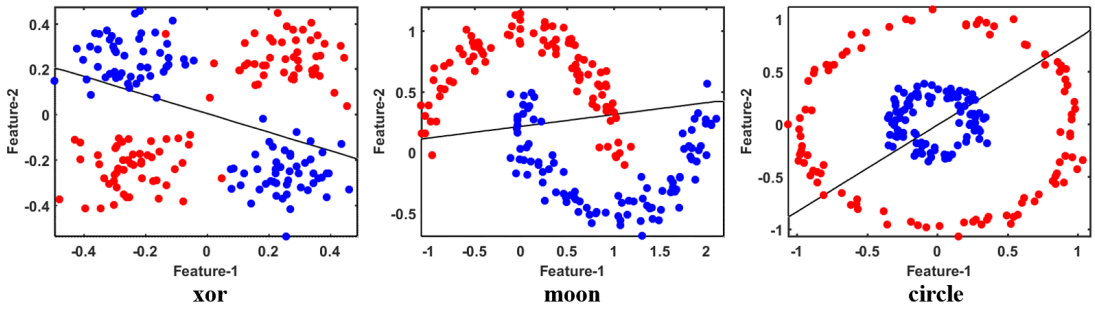


(b) Decision boundaries found by LDAC-CBDR

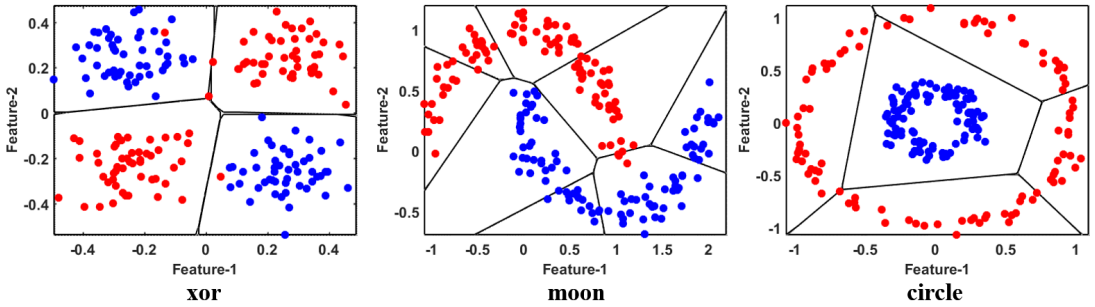


(c) Decision boundaries found by QDA

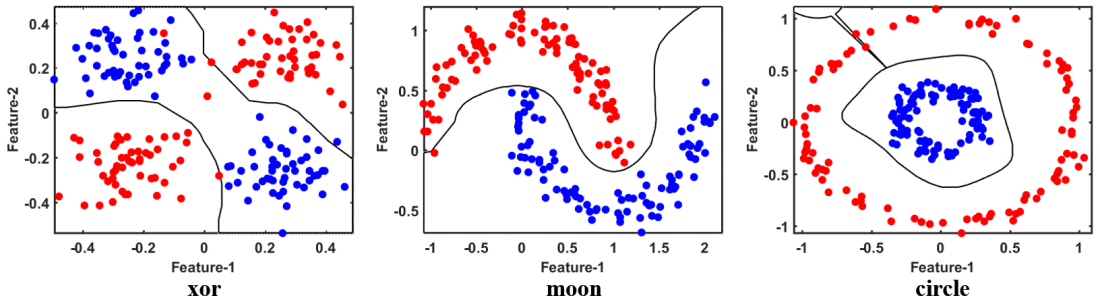
Figure 5.4: Decision boundaries found by LDAC, LDAC-CBDR and QDA, where differently coloured dots represent samples from different classes, and black straight lines represent decision boundaries.



(a) Decision boundaries found by LMLP

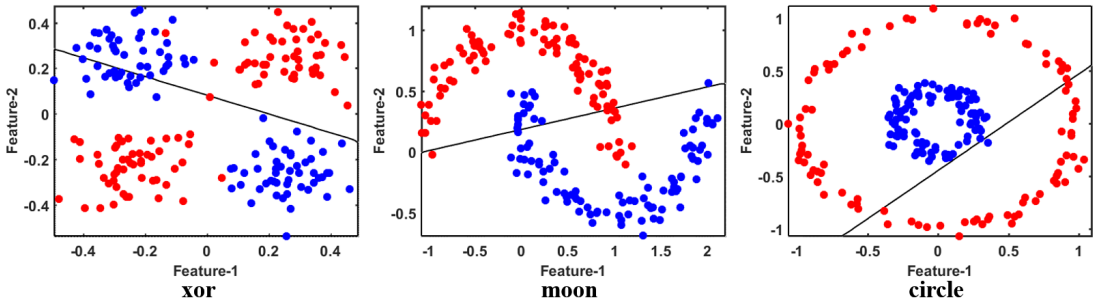


(b) Decision boundaries found by LMLP-CBDR

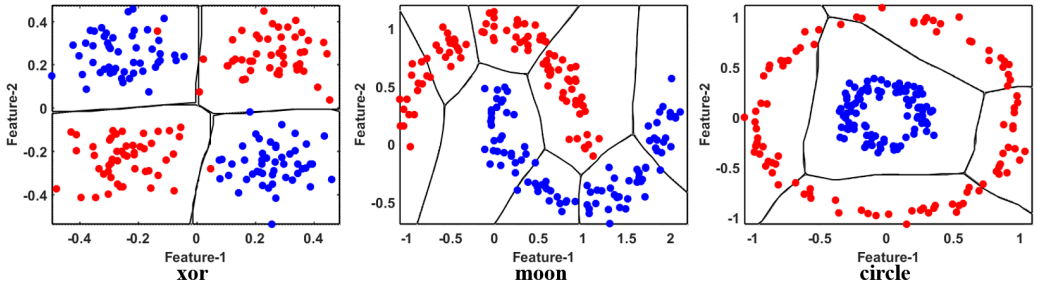


(c) Decision boundaries found by TSMLP

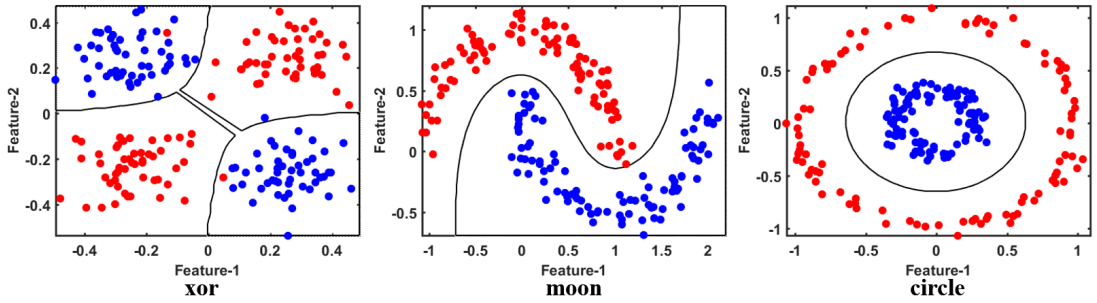
Figure 5.5: Decision boundaries found by LMLP, LMLP-CBDR and TSMLP, where differently coloured dots represent samples from different classes, and black straight or curved lines represent decision boundaries.



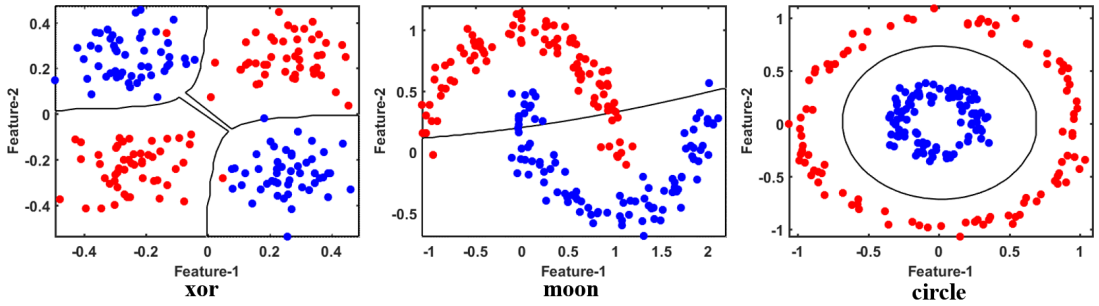
(a) Decision boundaries found by LSVM



(b) Decision boundaries found by LSVM-CBDR



(c) Decision boundaries found by RSVM



(d) Decision boundaries found by PSVM

Figure 5.6: Decision boundaries found by LSVM, LSVM-CBDR, RSVM and PSVM, where differently coloured dots represent samples from different classes, and black straight or curved lines represent decision boundaries.

Comparisons between linear classifiers equipped with CBDR and nonlinear classifiers: We first compare the decision boundaries found by linear classifiers equipped with CBDR against the ones found by nonlinear classifiers. It is easily observed from Figure 5.4(b-c), Figure 5.5(b-c) and Figure 5.6(b-d) that the decision boundaries found by the two types of classifier are quite different. Although the decision boundaries found by linear classifiers based on CBDR are straight lines, they achieve classification results that are similar to those obtained by the curved decision boundaries found by nonliear classifiers, and they are even better than some nonlinear classifiers. For example, the curved decision boundaries found by QDA and PSVM fail to separate well the two classes of the *moon* data set, but LDAC-CBDR and LSVM-CBDR do. Additionally, based on the classification accuracies shown in Figure 5.7, Figure 5.8 and Figure 5.9, we see that CBDR enhances the linear classifiers to be better or comparable to the nonlinear classifiers on nonlinear data.

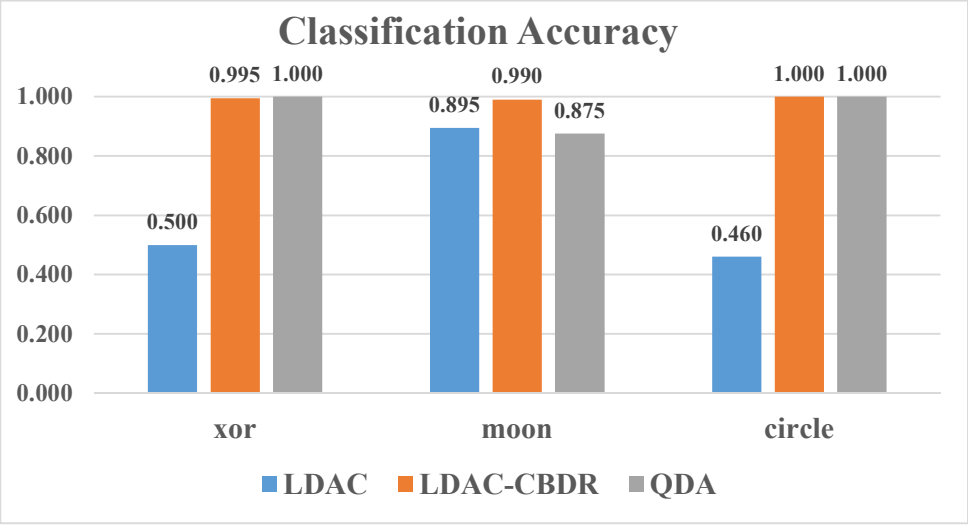


Figure 5.7: Classification accuracies of LDAC, LDAC-CBDR and QDA classifiers on *xor*, *moon* and *circle* data sets.

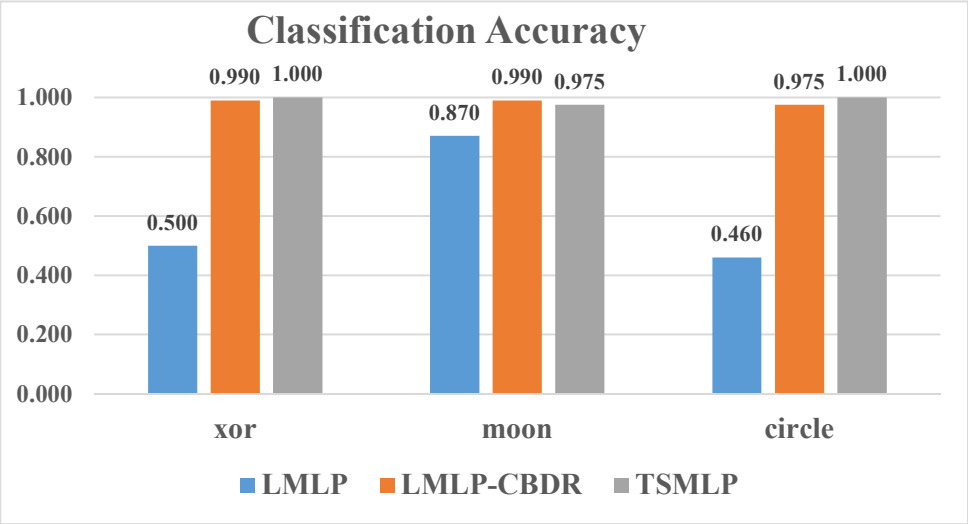


Figure 5.8: Classification accuracies of LMLP, LMLP-CBDR and TSMLP classifiers on *xor*, *moon* and *circle* data sets.

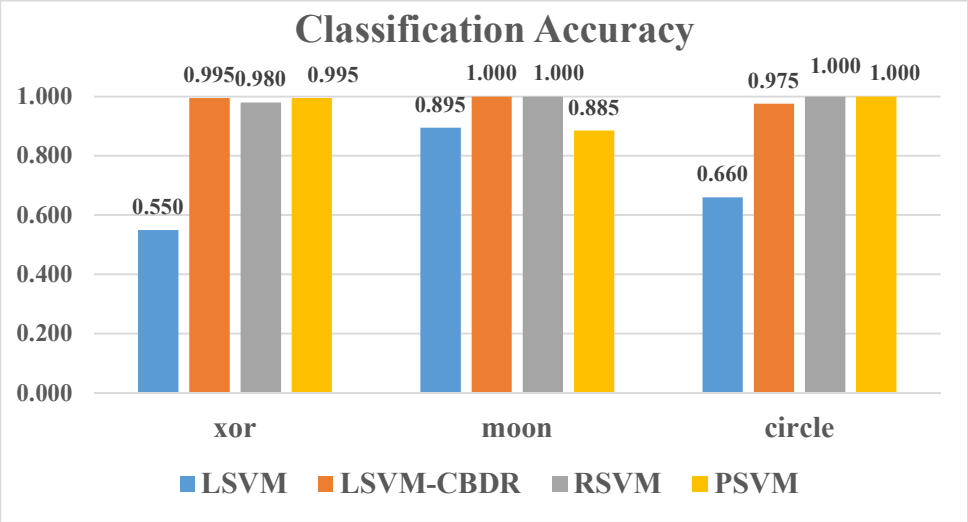


Figure 5.9: Classification accuracies of LSVM, LSVM-CBDR, RSVM and PSVM classifiers on *xor*, *moon* and *circle* data sets.

5.3.2 Evaluation Using Real-World Data

In this section, we evaluate the proposed CBDR on real-world data. Again we compare linear classifiers equipped with CBDR against linear classifiers alone and nonlinear classifiers. To test the strong generality of CBDR, furthermore, we combine CBDR with each of two inherently nonlinear classifiers: naive Bayes with kernel density estimation (NB) and Decision Tree (DT), resulting in NB-CBDR and DT-CBDR, respectively. We consider two classification tasks: imbalanced classification and general classification. For the imbalanced classification task, we select six imbalanced data sets from the KEEL repository [2]. Ten data sets are chosen from UCI Data Repository [28] for general classification. The criteria of our selection are: (1) all attributes/features must be numerical; (2) the number of features is small or moderate so that the study of the *nonlinear problem* is meaningful, since data in the high-dimensional feature space are more likely to be linearly separable. General information about the six imbalanced and ten UCI datasets is shown in Table 5.1 and Table 5.2, respectively. To make the evaluation results reliable, we use ten-fold cross-validation and *Estimated Mean Accuracy* (EMA) and *Standard Error of the Mean* (SEM) [49] as the evaluation metrics: EMA = $\frac{\sum_{i=1}^{10} p_i}{10}$, where p_i denotes the percentage of correct classification in the ith fold validation; SEM = $\frac{\sigma}{\sqrt{10}}$, where $\sigma = \sqrt{\frac{\sum_{i=1}^{10}(p_i - EMA)^2}{9}}$. So, the higher EMA and lower SEM, the better classification performance is.

Table 5.1: General information about the six imbalanced data sets used in the experiments. In this table, #Class denotes the number of classes, #Attribute denotes the number of attributes, #Sample is the number of samples and IR means imbalanced ratio.

Name of dataset	#Class	#Attribute	#Sample	IR
Glass1	2	9	214	1.82
Pima	2	8	768	1.87
Wisconsin	2	9	683	1.86
Vowel0	2	13	988	9.98
Autos	6	15	159	16.0
Penbased	10	16	1100	1.95

Table 5.2: General information about ten UCI datasets used in the experiments. In this table, #Class denotes the number of classes, #Attribute denotes the number of attribute and #Sample is the number of samples.

Name of dataset (acronyms)	#Class	# Attribute	#Sample
Musk version 1 (Musk)	2	166	476
SPECTF heart (SHeart)	2	44	267
Wisconsin diagnotic breast cancer (WDBC)	2	30	569
Banknote	2	4	1,372
Sonar	2	60	208
Haberman	2	3	306
Seeds	3	7	210
Statlog vechicle Sihouettes (SVS)	4	18	846
Glass	6	9	214
Urban land cover (ULC)	9	147	675

Experimental Results on Real-World Data: The EMA$\pm$SEM values of all classifiers are shown in Table 5.3, Table 5.4, Table 5.5 and Table 5.6. In addition, the best result among all algorithms is in bold type in all tables. Firstly, we compare linear classifiers equipped with CBDR against linear classifiers alone and their nonlinear variants. According to Table 5.3, Table 5.4, Table 5.5, we have the following observations:

- Linear classifiers equipped with CBDR obtain the best results on the majority of all data sets. In particular, LSVM-CBDR achieves the best results on 12 out of 16 data sets.

- From Table 5.3, it can be seen that LDAC-CBDR outperforms LDAC on 5 out 6 imbalanced data sets and 8 out of 10 UCI datasets, demonstrating that LDAC-CBDR is better than LDAC. For *Pima*, *Haberman* and *ULC*, LDAC-CBDR only finds one subclass for each class so that LDAC-CBDR obtains the same classification accuracy as LDAC. In addition, comparing LDAC-CBDR with QDA, it is easily observed that LDAC-CBDR is better than QDA on the majority of imbalanced and UCI data sets. Furthermore, LDAC-CBDR is significantly superior to QDA on some data sets. In particular, QDA only achieves 0.6405, 0.4771, 0.6700 and 0.5652 for *Glass1*, *Autos*, *Musk* and *Glass* data set, respectively. But, LDAC-CBDR obtains 0.7894, 0.6492, 0.8758 and 0.6870 for them, respectively.

- Based on the classification accuracy shown in Table 5.4, similar to LDAC-CBDR, LMLP-CBDR improves LMLP on the majority of imbalanced and UCI data sets. Additionally, comparing LMLP-CBDR with TSMLP, it is clear that LMLP-CBDR outperforms TSMLP on 5 out 6 imbalanced data sets. However, for UCI data sets, it is interestingly found that LMLP-CBDR is better than TSMLP on most of data sets with more than two classes and achieves similar classification accuracy as TSMLP on the UCI data sets with two classes.

- From Table 5.5, we find that LSVM-CBDR enhances the classification accuracy of LSVM on most data sets and makes significant improvement on more than two classes. For instance, LSVM-CBDR improves the classification accuracy of LSVM on *Autos* from 0.4279 to 0.5904 and *ULC* from 0.3689 to 0.6325. Moreover, we can note that LSVM-CBDR achieves better results than RSVM and PSVM on 12 out of 16 data sets and 13 out of 16 data sets, respectively; in particular, LSVM-CBDR outperforms RSVM on all imbalanced data sets.

Therefore, based on these observations on the real-world data sets, they further confirm that CBDR can enhance the classification accuracy of the linear classifier, especially for classifying imbalanced data sets.

Then we apply CBDR on two inherently nonlinear classifiers: NB and DT classifiers. We compare NB with NB-CBDR, and DT with DT-CBDR. From Table 5.6, it is easily seen that CBDR enhances classification accuracies of both NB and DT on the majority of imbalanced and UCI data sets. In particular, CBDR significantly improves the NB classifier. NB-CBDR increases the classification accuracies of NB on most data sets by over 5%, and NB-CBDR even outperforms NB on *Haberman* by over 27%.

In summary, we note the following observations:

- CBDR enabled three classical linear classifiers to work well on nonlinear data — significantly outperforming these linear classifiers on their own, with performance

Table 5.3: EMA±SEM values of LDAC, LDAC-CBDR and QDA on all imbalanced and UCI datasets

Methods / Datasets	LDAC	LDAC-CBDR	QDA
	EMA ± SEM	EMA ± SEM	EMA ± SEM
Imbalanced Data Sets			
Glass1	0.6359±0.0199	**0.7894±0.0297**	0.6405±0.0291
Pima	**0.7783±0.0234**	**0.7783±0.0234**	0.7472±0.0202
Wisconsin	0.9605±0.0069	**0.9620±0.0062**	0.9546±0.0083
Vowel0	0.9514±0.0026	0.9828±0.0043	**0.9980±0.0013**
Autos	0.5596±0.0362	**0.6492±0.0381**	0.4771±0.0562
Penbased	0.8800±0.0090	0.9573±0.0056	**0.9664±0.0043**
UCI Data Sets			
Musk	0.8109±0.0125	**0.8758±0.0122**	0.6700±0.0129
SHeart	0.7610±0.0230	0.7869±0.0188	**0.7942±0.0055**
WDBC	0.9561±0.0075	**0.9684±0.0057**	0.9579±0.0134
Banknote	0.9759±0.0050	0.9825±0.0038	**0.9832±0.0039**
Sonar	0.7498±0.0376	**0.8648±0.0229**	0.7555±0.0333
Haberman	0.7481±0.0181	**0.7580±0.0214**	0.7484±0.0174
Seeds	**0.9667±0.0102**	**0.9667±0.0102**	0.9429±0.0156
SVS	0.7966±0.0107	0.8191±0.0124	**0.8569±0.0101**
Glass	0.6530±0.0356	**0.6870±0.0197**	0.5652±0.0282
ULC	**0.8090±0.0098**	**0.8090±0.0098**	0.3362±0.0174

Table 5.4: EMA±SEM values of LMLP, LMLP-CBDR and TSMLP on all imbalanced and UCI datasets

Methods / Datasets	LMLP	LMLP-CBDR	TSMLP
	EMA ± SEM	EMA ± SEM	EMA ± SEM
Imbalanced Data Sets			
Glass1	0.6628±0.0305	**0.7433±0.0207**	0.7431±0.0213
Pima	0.7783±0.0242	**0.7888±0.0205**	0.7706±0.0224
Wisconsin	0.9663±0.0079	**0.9692±0.0074**	0.9678±0.0057
Vowel0	0.9737±0.0027	0.9737±0.0027	**0.9909±0.0028**
Autos	0.5538±0.0385	**0.5788±0.0261**	0.5517±0.0442
Penbased	0.8636±0.0121	**0.8745±0.0094**	0.8745±0.0112
UCI Data Sets			
Musk	0.8171±0.0159	0.8465±0.0165	**0.9159±0.0090**
SHeart	0.7755±0.0204	0.7865±0.0112	**0.8093±0.0201**
WDBC	0.9701±0.0053	0.9701±0.0053	**0.9702±0.0074**
Banknote	0.9811±0.0048	0.9905±0.0034	**0.9964±0.0016**
Sonar	0.7783±0.0367	0.8317±0.0228	**0.8890±0.0229**
Haberman	0.7414±0.0192	**0.7477±0.0228**	0.7351±0.0236
Seeds	0.9381±0.0214	**0.9476±0.0180**	0.9286±0.0216
SVS	**0.7552±0.0109**	**0.7552±0.0109**	0.7400±0.0154
Glass	0.6494±0.0309	**0.6632±0.0310**	0.6262±0.0288
ULC	0.7511±0.0195	0.7511±0.0195	**0.8074±0.0135**

Table 5.5: EMA±SEM values of LSVM, SVM-CBDR, RSVM and PSVM on all imbalanced and UCI datasets

Methods / Datasets	LSVM	LSVM-CBDR	RSVM	PSVM
	EMA ± SEM	EMA ± SEM	EMA ± SEM	EMA ± SEM
Imbalanced Data Sets				
Glass1	0.6403±0.0095	**0.7853±0.0254**	0.7706±0.0136	0.7249±0.0295
Pima	0.7757±0.0246	**0.7783±0.0236**	0.6509±0.0216	0.6534±0.0207
Wisconsin	**0.9678±0.0081**	**0.9678±0.0081**	0.8813±0.0116	0.9327±0.0085
Vowel0	0.9717±0.0036	**1.0000±0.0000**	0.9980±0.0013	**1.0000±0.0000**
Autos	0.4279±0.0551	**0.5904±0.0363**	0.2888±0.0335	0.2892±0.0210
Penbased	0.9709±0.0057	**0.9773±0.0045**	0.0691±0.0028	0.7027±0.0136
UCI Data Sets				
Musk	0.8363±0.0197	**0.9011±0.0185**	0.5652±0.0022	0.5551±0.0276
SHeart	0.7647±0.0225	0.7869±0.0231	**0.7942±0.0055**	0.7120±0.0181
WDBC	0.9525±0.0095	**0.9578±0.0095**	0.6274±0.0171	0.6274±0.0171
Banknote	0.9884±0.0036	**1.0000±0.0000**	**1.0000±0.0000**	0.9985±0.0010
Sonar	0.7926±0.0138	0.8167±0.0231	**0.8695±0.0152**	0.8457±0.0176
Haberman	0.7186±0.0190	**0.7611±0.0214**	0.7351±0.0213	0.5257±0.0496
Seeds	0.9095±0.0218	0.9143±0.0254	0.8857±0.0227	**0.9619±0.0138**
SVS	0.8227±0.0095	**0.8534±0.0103**	0.4066±0.0111	0.2045±0.0126
Glass	0.6448±0.0273	0.7015±0.0234	**0.7100±0.0172**	0.6872±0.0308
ULC	0.3689±0.0294	**0.6325±0.0186**	0.1466±0.0141	0.1154±0.0189

Table 5.6: EMA ±SEM values of NB, NB-CBDR, DT and DT-CBDR on all imbalanced and UCI datasets

Methods / Datasets	NB	NB-CBDR	DT	DT-CBDR
	EMA ± SEM	EMA ± SEM	EMA ± SEM	EMA ± SEM
Imbalanced Data Sets				
Glass1	0.7160±0.0324	**0.8136±0.0315**	0.7606±0.0338	**0.8132±0.0331**
Pima	0.7355±0.0171	**0.7614±0.0260**	0.7019±0.0191	**0.7121±0.0191**
Wisconsin	0.9678±0.0061	0.9678±0.0061	0.9502±0.0090	**0.9590±0.0075**
Vowel0	0.9767±0.0030	**0.9909±0.0035**	0.9818±0.0039	**0.9889±0.0024**
Autos	0.5733±0.0265	**0.6417±0.0307**	0.7863±0.0311	0.7863±0.0311
Penbased	0.8573±0.0129	**0.9391±0.0043**	0.8636±0.0125	**0.8945±0.0136**
UCI Data Sets				
Musk	0.8319±0.0137	**0.9095±0.0115**	0.7668±0.0218	**0.8512±0.0247**
SHeart	0.7303±0.0302	**0.8017±0.0193**	0.7604±0.0247	**0.7868±0.0242**
WDBC	0.9438±0.0100	0.9438±0.0100	0.9192±0.0131	**0.9579±0.0047**
Banknote	0.9176±0.0038	**0.9789±0.0054**	0.9854±0.0029	0.9854±0.0029
Sonar	0.7788±0.0146	**0.8357±0.0214**	0.7164±0.0249	**0.7929±0.0194**
Haberman	0.4600±0.0480	**0.7382±0.0184**	0.6731±0.0263	**0.7022±0.0219**
Seeds	0.9143±0.0138	**0.9238±0.0162**	0.9286±0.0177	0.9286±0.0177
SVS	0.6205±0.0125	**0.7092±0.0115**	0.7506±0.0189	**0.7540±0.0161**
Glass	0.6543±070343	**0.7097±0.0148**	0.7006±0.0257	**0.7106±0.0234**
ULC	0.8236±0.0151	**0.8369±0.0115**	0.7733±0.0107	**0.7895±0.0144**

that is competitive with or even better their nonlinear variants.

- CBDR improved performance of two additional inherent nonlinear classifiers.
- In particular, CBDR led to consistent and significant outperformance on imbalanced data.

5.4 Summary

In this chapter, we have proposed a general method, *cluster-based data relabelling* (CBDR), to improve the classification performance of linear classifiers as well as nonlinear ones, and demonstrated its achievement of this goal through extensive experimentation using a large number of classifiers on a wide range of artificial and real data sets. CBDR seeks to find non-overlapping class-specific clusters, relabel data by the clusters they belong to in order to learn cluster-based decision boundaries rather than class-based decision boundaries. A mapping between new class labels and original ones can be easily set up since each cluster belongs to one and only one class. A linear classifier, in fact any classifier, can be applied to the relabelled data. Experiments on artificial nonlinear data sets (xor, moon and circle) have illustrated the cluster-based decision boundaries and have demonstrated superior classification performance by CBDR. Further extensive experiments on real data sets have demonstrated significant performance gains due to the use of CBDR. These experiments provide strong evidence for us to conclude that CBDR is an effective generic method for enhancing the classification performance of linear classifiers as well as nonlinear classifiers. In particular, this method could potentially offer an excellent solution to the class imbalanced classification problem.

CHAPTER VI

A Novel Gaussian Mixture Model for Classification

To further demonstrate the value of cluster-based supervised classification, we present a novel extension of the original Gaussian mixture model (GMM) classifier, called *separability criterion based GMM* (SC-GMM) classifier.

6.1 Motivation

Gaussian mixture models are a probabilistic model for representing normally distributed subpopulations within an overall population. Mixture models in general do not require knowledge of which subpopulation a data point belongs to, allowing the model to learn the subpopulations automatically. Since subpopulation assignment is not known, this constitutes a form of unsupervised learning.

In statistics and machine learning, classification is the problem of assigning a data sample to one of a group of categories based on the information available in a set of data samples whose class labels are known. Classification is studied in many areas, including pattern recognition, computer vision and natural language processing. Many methods have been proposed to learn to perform classification. Some aim to separate different classes by maximising the margins of decision boundaries between classes, for example SVM [20]. Some methods utilise Bayes decision theory to calculate class posterior probabilities from class prior probabilities and class-conditional densities. In the real

world, however, the boundaries between different classes are complex, and the estimates of the class-conditional densities may not be accurate enough. This may be caused by a lack of information about the true functional form of the class-conditional density or an insufficient number of samples that can be used to estimate the parameters of the class-conditional densities [81]. Fortunately, mixture models can be used to represent arbitrarily complex probability density functions [32], which makes mixture models a suitable candidate to represent complex class-conditional densities. In particular, it has been demonstrated that Gaussian Mixture Model (GMM) can approximate any target density with arbitrary precision [72]. GMM is usually used for unsupervised learning, but is also used for supervised learning or classification. However, the performance of GMM as a classifier is not impressive compared with other conventional classifiers such as k-nearest neighbours, support vector machine, decision tree and naive Bayes.

In order to improve the performance of GMM as a classifier, we need to answer an important question. How many Gaussian components should be used to approximate the data distribution? A GMM with too many Gaussian components may overfit the data, whilst a GMM with too few components may not be flexible enough to approximate the true underlying density distribution. Therefore, it is necessary for a GMM classifier to choose the optimal number of Gaussian components for different data sets.

In this chapter, we seek to answer the above question to improve GMM in its classification performance. We propose a GMM classifier, SC-GMM, based on a separability criterion, which can automatically find the optimal number of Gaussian components for a given data set. The separability criterion [119] aims to find the clustering of each class that maximises the average distance between the mean of a class and the means of clusters in this class. So, the clusters found by the separability criterion for each class are mutually as separate as possible, like the clusters shown in Figure 6.1. SC-GMM takes the number of clusters as the optimal number of Gaussian components for each class, and uses it as the input to the Expectation-Maximisation (EM) algorithm [23]

to obtain the final GMM. This final GMM is our SC-GMM classifier.

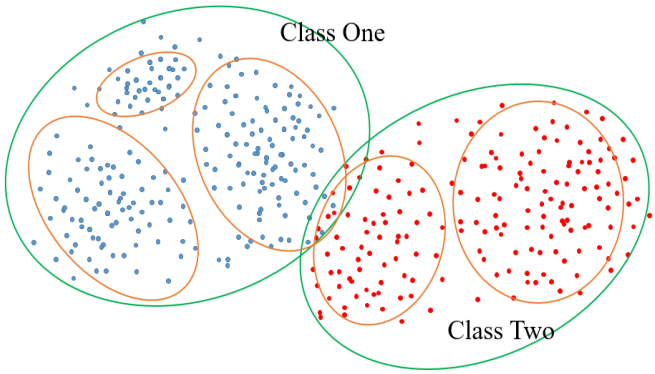


Figure 6.1: Illustration of clusters found by the separability criterion, where the orange circles in each class represent clusters.

The remainder of this chapter is organised as follows. Section 6.2 presents related work, which briefly introduces GMM and the selection of GMM. The details of the SC-GMM classifier are described in Section 6.3. Experiments on a set of machine learning data sets are provided in Section 6.4. Finally, Section 6.5 concludes this chapter.

6.2 Related Work

6.2.1 Gaussian Mixture Model

A Gaussian Mixture Model (GMM) is a category of probabilistic model which states that all generated data samples are derived from a mixture of finite Gaussian densities. So, GMM models the distribution of a data set by using a number of Gaussian densities. In general, a Gaussian density in a *d*-dimensional space is defined as:

$$N(\mathbf{x}|\mu,\Sigma) = \frac{1}{(2\pi)^{d/2}(|\Sigma|)^{1/2}} \exp(-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)), \tag{6.1}$$

where $\mathbf{x}$ represents a *d*-dimensional data vector, $\mu \in R^d$ is the mean of a Gaussian density and Σ is the $d \times d$ covariance matrix of the Gaussian density. In GMM, a Gaussian density

$N(\mathbf{x}|\mu, \Sigma)$ is usually called a component. So, we can describe a K-component GMM as:

$$f_K(\mathbf{x}) = \sum_{k=1}^{K} w_k N(\mathbf{x}|\mu_k, \Sigma_k), \tag{6.2}$$

where w_k is the mixing weight of the kth component, with $\sum_{k=1}^{K} w_k = 1$ and $w_k \geq 0, k = 1, 2, \cdots, K$. According to Equation (6.2), if the mean μ_k and covariance matrix Σ_k of components as well as mixing weights w_k are known, then we can easily model the distribution of a data set by using GMM. Unfortunately, w_k, μ_k and Σ_k are unknown in most cases. So, w_k, μ_k and Σ_k are the parameters of GMM, and we use the notation θ to collectively denote the parameters of GMM as: $\theta = \{w_k, \mu_k, \Sigma_k\}, k = 1, 2, \cdots, K$.

To estimate θ, the well known Expectation-Maximisation (EM) algorithm [23] is commonly used. The EM algorithm is a maximum likelihood estimation method, which finds the maximum likelihood of a model through iteration. There are two steps in the EM algorithm: expectation step and maximisation step. The EM algorithm alternates between performing an expectation step and a maximisation step to achieve the maximum likelihood until a certain stop condition is satisfied or a specified number of iterations is completed.

For a given K-component GMM with respect to a data set $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$ and all $\mathbf{x}_i \in R^d$, the algorithm is given as follows:

- Expectation step: the EM algorithm calculates the likelihoods that the components generate each sample. The likelihood $L(C_k|\mathbf{x}_i)$ that the kth component C_k generates sample $\mathbf{x}_i$ can be obtained by:

$$L(C_k|\mathbf{x}_i) = \frac{w_k N(\mathbf{x}_i|\mu_k, \Sigma_k)}{\sum_{k=1}^{K} w_k N(\mathbf{x}_i|\mu_k, \Sigma_k)}. \tag{6.3}$$

- Maximisation step: the EM algorithm updates the mean and covariance of each

component as well as the mixing weights based on the likelihoods calculated in the expectation step as follows:

$$w_k = \frac{\sum_{i=1}^{n} L(C_k|\mathbf{x}_i)}{n}, \tag{6.4}$$

$$\mu_k = \frac{\sum_{i=1}^{n} L(C_k|\mathbf{x}_i)\mathbf{x}_i}{nw_k}, \tag{6.5}$$

$$\Sigma_k = \frac{\sum_{i=1}^{n} L(C_k|\mathbf{x}_i)(\mathbf{x}_i - \mu_k)(\mathbf{x}_i - \mu_k)^T}{nw_k}, \tag{6.6}$$

where n is the number of samples. Through the iterative application of Equation (6.3) and Equations (6.4), (6.5) and (6.6) for all components C_k, we complete the estimation of the parameters θ.

6.2.2 Gaussian Mixture Model Selection

For a given set of data which includes classes $c_i, i = 1, 2, \cdots, C$, the task of a classifier is to assign a sample $\mathbf{x}$ to one of the C classes. Based on Bayes decision theory, which aims to minimise the probability of error, the classifier assigns $\mathbf{x}$ to the class c_i that yields the maximum posteriori probability $P(c_i|\mathbf{x})$. Usually, $P(c_i|\mathbf{x})$ is unknown, but according to Bayes rule, $P(c_i|\mathbf{x})$ can be calculated by using the class-conditional probability $P(\mathbf{x}|c_i)$ and the prior probability of the class $P(c_i)$. So, the main idea of a classifier based on Bayes decision theory can be stated as:

$$\mathbf{x} \in c_i \;\; if \;\; c_i = \arg\max_{c_j} P(\mathbf{x}|c_j)P(c_j). \tag{6.7}$$

As a classifier that is based on Bayes decision theory, the GMM classifier uses several Gaussian distributions to approach the true density of a class. It obtains class-conditional probability based on these Gaussian distributions, and then computes posteriori probability.

How many Gaussian distributions should be used for each class? There are many GMM selections for every class. So, we assume that there are L_i candidate Gaussian mixture models for class c_i and denote them as $M_{il}, l = 1, 2, \cdots, L_i$. Here, each M_{il} consists of the parameters of the model θ_{il} and the number of Gaussian components K_{il}. Thus, each model M_{il} implements the class-conditional probability $P(\mathbf{x}|c_i)$ as $\sum_{k=1}^{K_{il}} P(\mathbf{x}|\theta_{il}, k)$. The best model M_{ij} is selected to model the data distribution of class c_i, when it satisfies the following condition:

$$M_{ij} = \arg\max_{l} P(c_i) \sum_{k=1}^{K_{il}} P(\mathbf{x}|\theta_{il}, k). \tag{6.8}$$

Therefore, the selection of the Gaussian mixture model is significant for the GMM classifier, which includes the selection of θ_{il} and K_{il}. According to the description of GMM above, the parameter θ_{ij} of M_{ij} can be estimated by the EM algorithm. But, how do we find the number of Gaussian components K of model M_{ij}? This is a big problem for a GMM classifier, as the mixture may overfit the data if there are too many components, whilst a mixture with too few components may not be flexible enough to approximate the true underlying model. So, the selection of K significantly influences the performance of the GMM classifier. There are many criteria proposed for GMM to select K: the well-known and commonly used criteria are Akaike information criterion (AIC) [1] and Bayesian information criterion (BIC) [106].

Both AIC and BIC select the optimal value for K by introducing a penalty term to find the trade-off between the goodness of fit of the model and the simplicity of the model. They are respectively defined as:

$$S_{AIC} = 2p - 2ln(\hat{L}) \tag{6.9}$$

$$S_{BIC} = pln(n) - 2ln(\hat{L}) \tag{6.10}$$

where $ln()$ is the logarithm function, p is the number of estimated parameters in the model, $\hat{L}$ is the maximum value of the likelihood function for the model, and n denotes the number of samples in a data set. AIC takes the number of components of the model

which minimises S_{AIC} as the optimal number of components K_{opt}. Similarly to AIC, BIC prefers the model with the lowest S_{BIC} and takes the corresponding number of components as K_{opt}.

The algorithm based on AIC or BIC to find the optimal number of components K_{opt} for each class is summarised in Algorithm 4. Given a maximum value K_{max}, we run the EM algorithm K_{max} times with $k = 1$ to K_{max}, calculating S_{AIC} or S_{BIC} for every k and then take the k corresponding to the lowest value of S_{AIC} or S_{BIC} as the optimal value K_{opt}.

Algorithm 4 Finding the optimal number of components K_{opt} based on the AIC or BIC criterion. In this algorithm, C is the number of classes, K_{max} denotes the maximum number of Gaussian components to consider for a class and K^i_{opt} is the optimal number of Gaussian components of the ith class.

Input: A set of training samples T_{set} and K_{max}.

Output: K_{opt}.

$K_{opt} = \{\}$;
for $i = 1$ to C **do**
    for $k = 1$ to K_{max} **do**
        Apply GMM with k on T_{set}.
        $p, \hat{L} = GMM(T_{set}, k)$;
        $S_{AIC} = 2p - 2ln(\hat{L})$ or $S_{BIC} = pln(n) - 2ln(\hat{L})$;
    end for
    $K^i_{opt} = \arg\min_k (S_{AIC} \ \ or \ \ S_{BIC})$;
    $K_{opt} = K_{opt} \cup \{K^i_{opt}\}$;
end for
return K_{opt}

Based on the values of K_{opt} found by AIC or BIC as well as the EM algorithm, we can obtain the final GMMs of a data set. We denote the final GMMs as AIC-GMM classifier and BIC-GMM classifier, respectively.

6.3 Separability Criterion for GMM Classifier

6.3.1 SC-GMM Classifier

In this section, we give the details of the GMM classifier based on a separability criterion, which is called *SC-GMM classifier*. SC-GMM classifier uses the separability criterion [119] and an agglomerative hierarchical clustering algorithm to find K_{opt} for each class. The separability criterion attempts to find the clustering that maximises the average distance between the mean of a class and the means of clusters in this class, which is defined as follows:

$$K_i^* = \arg\max_k (AED_{ik}), \tag{6.11}$$

where K_i^* is the optimal number of clusters in class i. We take K_i^* as the optimal number of Gaussian components K_{opt} for class i, and AED_{ik} is the *average Euclidean distance* (AED) between the mean of class i and the means of its k clusters.

$$AED_{ik} = \frac{1}{k}\sum_{j=1}^{k}\|\mu_{ij} - \mu_i\|_2^2, \tag{6.12}$$

where μ_{ij} is the mean of the jth cluster in class i and μ_i is the mean of class i. It is clear that the larger AED, the more separated the clusters are from each other. The algorithm based on the separability criterion to find K_{opt} for each class is described in Algorithm 5.

After obtaining the values of K_{opt}, we use these K_{opt} values and the EM algorithm to obtain the final GMM of a data set. The final GMM is the SC-GMM classifier. Apart from the AIC-GMM classifier and BIC-GMM classifier, we also compare SC-GMM classifier with variational Bayesian Gaussian mixture (VBGM). VBGM is a state-of-the-art variant of the GMM with variational inference algorithms [5] that maximises a lower bound on model evidence instead of data likelihood. To get the values of K_{opt} for each class, VBGM uses the Dirichlet process inference algorithm (DP) [9].

Algorithm 5 Finding the optimal number of components K_{opt} based on the separability criterion. In this algorithm, *C* is the number of classes, μ_i is the mean of the ith class, μ_{ik} is the mean of the kth cluster in the ith class, K_{max} denotes the maximum number of components to consider for a class, and K^i_{opt} is the optimal number of components of the ith class.

Input: A set of training samples T_{set} and K_{max}.

Output: K_{opt}.

$K_{opt} = \{\}$
for $i = 1$ to C **do**
    $AED_i = \{\}$
    Calculate μ_i.
    for $k = 1$ to K_{max} **do**
        Apply agglomerative hierarchical clustering (HC) and obtain $Clusters$:
        $Clusters = HC(T_{set}, k)$;
        Calculate μ_{ik} based on $Clusters$.
        $AED_{ik} = \frac{1}{k}\sum_{j=1}^{k}\|\mu_{ij} - \mu_i\|_2^2$;
        $AED_i = AED_i \cup \{AED_{ik}\}$;
    end for
    $K^i_{opt} = \arg\max_k (AED_i)$.
    $K_{opt} = K_{opt} \cup \{K^i_{opt}\}$
end for
return K_{opt}

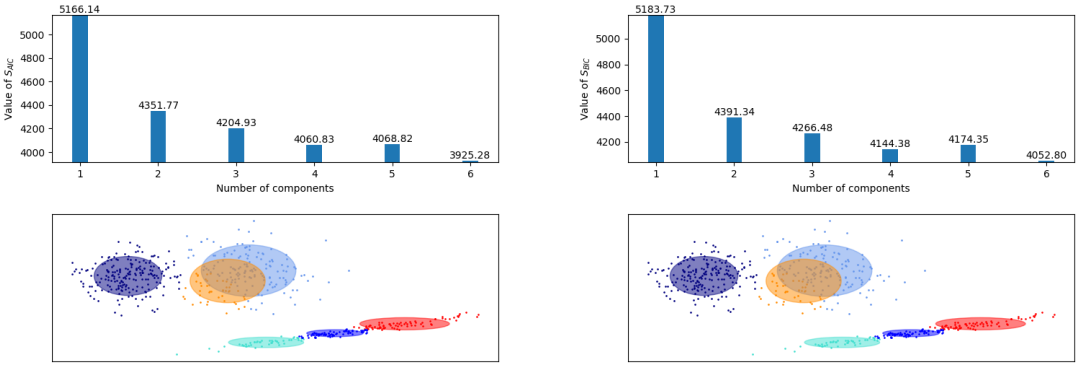
6.3.2 Comparison of AIC-GMM Classifier, BIC-GMM Classifier, SC-GMM Classifier and VBGM Classifier

Firstly, we compare the selected GMMs separately based on AIC, BIC, SC criteria and DP on an artificial dataset. The artificial dataset is composed of data from three normal distributions. The details of the three normal distributions are shown in the Table 6.1. We apply Algorithm 4 and Algorithm 5 to the artificial dataset. We set $C = 1$ and $K_{max} = 6$. A visualisation of the selected GMM based on AIC, BIC, SC and DP is shown in Figure 6.2. From Figure 6.2 (a) and (b) we see that S_{AIC} and S_{BIC} achieve the lowest values when $K_{opt} = 6$, so both AIC-GMM classifier and BIC-GMM classifier for the artificial dataset are composed of six components. In contrast, SC-GMM classifier consists of four components as the SC criterion achieves the largest AED when $K_{opt} = 4$. In addition, VBGM also finds four components for the artificial data based on DP. Comparing these GMMs for the artificial dataset, we find that the GMMs of the SC-GMM classifier and VBGM classifier are closer to the actual distribution of the artificial data.

Table 6.1: Details of the artificial data used

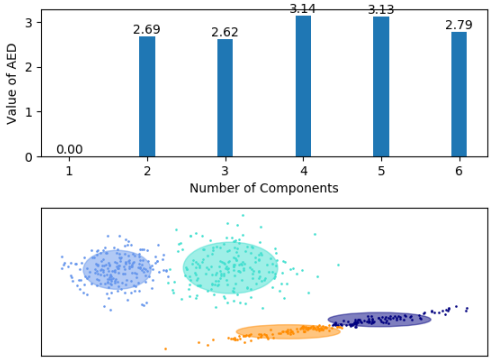
Normal distribution	Means	Covariance	# Instances	# Attributes
N_1	[0 -0.1]	$\begin{bmatrix} 1.7 & 0 \\ 0 & 0.4 \end{bmatrix}$	200	2
N_2	[-3.0 3.0]	$\begin{bmatrix} 1.0 & 0 \\ 0 & 1.0 \end{bmatrix}$	200	2
N_3	[-6.0 3.0]	$\begin{bmatrix} 0.7 & 0 \\ 0 & 0.7 \end{bmatrix}$	200	2

We also compare the computational times that AIC-GMM, BIC-GMM and SC-GMM classifiers spend on determining K_{opt}. According to Algorithm 4 and Algorithm 5, we can calculate their computational complexities as $C * K_{max} * t * O(k_i n_i D^2)$ and $C * K_{max} * O(n_i^2 log n_i)$, respectively. Here, t is the number of iterations of the EM algorithm in GMM, $O(k_i n_i D^2)$ is the time expense per iteration of the EM algorithm [88], k_i is the number of components of class i, n_i is the number of samples of class i, D is the dimension of the

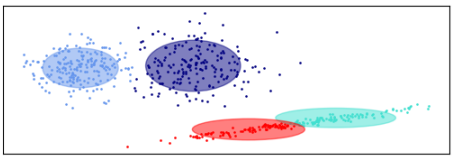


(a) Selected GMM based on AIC criterion

(b) Selected GMM based on BIC criterion



(c) Selected GMM based on SC criterion



(d) Selected GMM based on DP

Figure 6.2: Visualisations of selected GMM based on AIC, BIC, SC criteria and DP on the artificial data set. Circles with different colours denote different components

sample, and $O(n_i^2 log n_i)$ is the time complexity of the agglomerative hierarchical clustering algorithm [11]. We note that the increase in time complexity of Algorithm 5 is only affected by n_i except C and K_{max}, whereas Algorithm 4 is influenced by the growth of t, k_i, n_i and particularly D. Therefore, the AIC-GMM classifier and the BIC-GMM classifier both need much more time to find K_{opt} for each class than the SC-GMM classifier, especially for data sets with high dimensionality, such as a face dataset. Furthermore, extensive experiments on real data sets confirm this, as discussed in Section 6.4.2.

6.4 Experiments

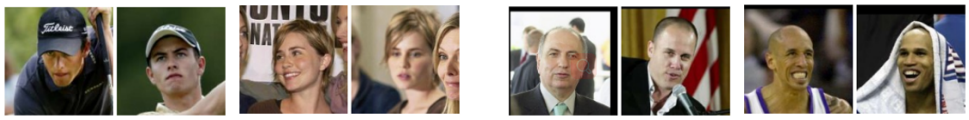
In our experiments we consider two classification tasks to evaluate the SC-GMM classifier: general data mining and face verification. For general data mining, we conduct experiments on ten data sets from the UCI machine learning repository [74] and the KEEL Data Repository [3][2]. The details of the ten data sets are shown in Table 6.2. For face verification, we use face images from the Labeled Faces in the Wild (LFW) face database [49].

Table 6.2: General information about the ten data sets used

Name of data set (Acronym)	#Instances	#Classes	#Attributes
Climate	540	2	18
Dermatology	366	6	34
Glass1	214	2	9
Ionosphere	351	2	33
Musk(Version 1)-Clearn1 (M1-C1)	476	2	166
Seeds	210	3	7
Sonar	208	2	60
Spectheart	267	2	44
Hayes-roth	132	3	4
Multi-feature-fou(MF-fou)	2000	10	76

Face verification is a binary classification problem that aims to decide whether two given face images belong to the same person (matched pair) or not (mismatched pair). LFW is an important benchmark database in the face verification area. It comprises 13,233 face images of 5,749 people, which are collected from the Internet with large intra-personal variations (see Figure 6.3 for some examples). The LFW database is grouped into two views. View One is a development set of 3,200 pairs, which is used to build models and select features. View Two is a ten-fold cross-validation set of 6,000 pairs for evaluation. The size of each image is 250 by 250 pixels. There are three versions of LFW: original, funneled and aligned. In our experiments, we use the aligned version [126] and a subset of View Two of LFW. We randomly select 200 matched face pairs and 200 mismatched face pairs from View Two and crop each image to a size of 80 by 150 pixels as in [58]. Hence, we have 24,000 features for each image of LFW. Also, we implement dimensionality reduction

for face images before using the classifiers. We use PCA to reduce data dimensionality and retain principal components that can explain 95 percent of the variance.



(a) Examples of matched pairs (b) Examples of mismatched pairs

Figure 6.3: Sample images from the LFW face databases

We verify the classification performance of the SC-GMM classifier from three perspectives: 1) comparing SC-GMM with the original GMM classifier, which does not use any criteria to find the optimal number of Gaussian components for each class and just models each class using one Gaussian component; 2) comparing SC-GMM with AIC-GMM, BIC-GMM and VBGM; 3) comparing SC-GMM with five other widely used classifiers, namely *k*-nearest neighbours (*k*NN), linear support vector machine (Linear-SVM), radial basis function kernel SVM (RBF-SVM), decision tree (DT) and naive Bayes (NB). All of these five classifiers can be found in Scikit-learn [96]. In addition, in our experiments we use ten-fold cross-validation as the evaluation framework and take *estimated mean accuracy* (EMA) and *standard error of the mean* (SEM) as evaluation metrics. EMA and SEM are defined as follows: $EMA = \frac{\sum_{i=1}^{10} p_i}{10}$, where p_i denotes the percentage of correct classification in the ith fold of validation; $SEM = \frac{\delta}{\sqrt{10}}$, where $\delta = \sqrt{\frac{\sum_{i=1}^{10}(p_i - EMA)^2}{9}}$.

6.4.1 SC-GMM vs GMM

Firstly, we compare the SC-GMM classifier with the original GMM classifier on the ten general data sets and LFW face database. The classification accuracies of the original GMM and SC-GMM classifiers are shown in Table 6.3. From Table 6.3 it is clear that SC-GMM achieves higher classification accuracy than the original GMM on all ten data sets. Furthermore, SC-GMM dramatically improves the classification performance of the original GMM on most of the data sets. For example, SC-GMM improves EMA: 0.5682

of original GMM to EMA: 0.7955 on Glass1 data set, an increase of 22.73%, and improves the classification accuracy of GMM on Sonar data set from EMA: 0.6905 to EMA: 0.8619, an increase of 17.14%. For the face verification task, the SC-GMM classifier also has superior verification performance to the original GMM. These observations show that it is insufficient to model the data distribution of a class with only one Gaussian component. Furthermore, the classification performance of the GMM classifier has been improved significantly through modelling the data distribution of every class using several Gaussian components.

Table 6.3: The classification accuracies of original GMM and SC-GMM on all data sets

Methods / Data sets	original GMM	SC-GMM
	EMA $\pm$ SEM	EMA $\pm$ SEM
Climate	0.8870 $\pm$ 0.0384	**0.9259 $\pm$ 0.0406**
Dermatology	0.8973 $\pm$ 0.0495	**0.9622 $\pm$ 0.0367**
Glass1	0.5682 $\pm$ 0.1483	**0.7955 $\pm$ 0.0547**
Ionosphere	0.8944 $\pm$ 0.0553	**0.9250 $\pm$ 0.0431**
M1-C1	0.7500 $\pm$ 0.0618	**0.8854 $\pm$ 0.0429**
Seeds	0.8857 $\pm$ 0.0646	**0.9333 $\pm$ 0.0680**
Sonar	0.6905 $\pm$ 0.1300	**0.8619 $\pm$ 0.0655**
Spectheart	0.6741 $\pm$ 0.0737	**0.7778 $\pm$ 0.0794**
Hayes-roth	0.6571 $\pm$ 0.1187	**0.7643 $\pm$ 0.1062**
MF-fou	0.7500 $\pm$ 0.0257	**0.8150 $\pm$ 0.0266**
LFW	0.6250 $\pm$ 0.0750	**0.6450 $\pm$ 0.0678**

6.4.2 SC-GMM vs AIC-GMM, BIC-GMM and VBGM

In this subsection SC-GMM is compared with AIC-GMM, BIC-GMM and VBGM. According to the classification accuracy shown in the Table 6.4, it can be seen that the SC-GMM classifier and the AIC-GMM classifier obtain close classification accuracy on all data sets, and SC-GMM outperforms both BIC-GMM and VBGM on 7 out of 11 data sets. In particular, SC-GMM achieves best face verification performance. Additionally, we compare SC-GMM with AIC-GMM and BIC-GMM in the speed of finding the optimal number of Gaussian components. Based on the computation time shown in Table 6.5, it is clear from this table that the separability criterion used in the SC-GMM classifier is faster than the AIC criterion used in AIC-GMM and BIC criterion used in BIC-GMM on all data

sets, which further confirms the abovementioned analysis of time complexity. Moreover, for the data set with high dimensionality, AIC-GMM and BIC-GMM are time-consuming. For example, the computation time of AIC-GMM and BIC-GMM on the LFW data set is 47.1232 seconds and 17.6830 seconds, respectively. In contrast, SC-GMM only consumes 0.4886 seconds. Hence, we can conclude that the SC-GMM is more efficient than the AIC-GMM classifier and BIC-GMM classifier.

Table 6.4: The classification accuracy of AIC-GMM, BIC-GMM, SC-GMM and VBGM on all data sets

Methods / Data sets	AIC-GMM	BIC-GMM	SC-GMM	VBGM
	EMA ± SEM	EMA ± SEM	EMA ± SEM	EMA ± SEM
Climate	0.9222 ± 0.0560	0.9167 ± 0.0334	0.9259 ± 0.0406	**0.9481 ± 0.0329**
Dermatology	**0.9649 ± 0.0272**	**0.9649 ± 0.0272**	0.9622 ± 0.0367	0.9459 ± 0.0342
Glass1	0.7727 ± 0.0643	0.7636 ± 0.0698	**0.7955 ± 0.0547**	0.7045 ± 0.1412
Ionosphere	**0.9278 ± 0.0377**	**0.9278 ± 0.0377**	0.9250 ± 0.0431	0.9167 ± 0.0448
M1-C1	**0.8937 ± 0.0342**	**0.8937 ± 0.0342**	0.8854 ± 0.0429	0.8229 ± 0.0504
Seeds	0.9381 ± 0.0641	0.9238 ± 0.0883	0.9333 ± 0.0680	**0.9524 ± 0.0563**
Sonar	**0.8667 ± 0.0513**	0.8381 ± 0.0680	0.8619 ± 0.0655	0.7571 ± 0.0837
Spectheart	0.7704 ± 0.0569	0.7704 ± 0.0569	0.7778 ± 0.0794	**0.7852 ± 0.0679**
Hayes-roth	0.7643 ± 0.1239	0.7643 ± 0.1239	0.7643 ± 0.1062	**0.8143 ± 0.0795**
MF-fou	**0.8165 ± 0.0258**	0.8135 ± 0.0182	0.8150 ± 0.0266	0.7485 ± 0.0258
LFW	0.6275 ± 0.0647	0.6275 ± 0.0728	**0.6450 ± 0.0678**	0.6275 ± 0.0596

Table 6.5: The computation time to find the optimal number of components by using AIC, BIC and SC criteria on all data sets (unit: seconds)

Methods / Data sets	AIC	BIC	SC
Climate	37.0739	21.0488	**3.8669**
Dermatology	10.8137	10.8571	**0.3709**
Glass1	10.2785	6.8076	**0.2877**
Ionosphere	5.3044	5.5403	**0.6093**
M1-C1	42.3008	42.7811	**2.3338**
Seeds	9.9542	2.1960	**0.2048**
Sonar	6.7193	8.5999	**0.2048**
Spectheart	26.3953	26.5191	**0.8569**
Hayes-roth	26.0510	26.8905	**0.2479**
MF-fou	71.2184	68.7672	**3.6695**
LFW	47.1232	17.6830	**0.4886**

6.4.3 SC-GMM vs Five Other Classical Classifiers

Finally, we conduct experiments to compare SC-GMM with five widely used classifiers: kNN(k=1), Linear-SVM, RBF-SVM, DT and NB on all data sets. The classification accuracy of different classifiers are shown in Table 6.6, and the best classification accuracy is in bold. Based on the comparison of classification accuracy shown in Table 6.6, we observe that:

- Separately comparing the SC-GMM classifier with kNN, Linear-SVM, RBF-SVM and NB, the SC-GMM classifier achieves higher classification accuracy on 8 out of 11 data sets, 10 out of 11 data sets, 8 out of 11 data sets and 10 out 11 data sets, respectively. Moreover, the SC-GMM classifier outperforms DT on all eleven data sets.

- Collectively comparing the SC-GMM classifier with the five classifiers, it is observed from Table 6.6 that the SC-GMM classifier obtains the best classification accuracy on 4 out of 11 data sets and second best on 7 out of 11 data sets, whilst kNN, Linear-SVM, RBF-SVM, DT and NB are only the best on 2 out of 11, 1 out of 11, 3 out of 11, 0 out of 11 and 1 out of 11, respectively. Moreover, the second best classification accuracy achieved by the SC-GMM classifier is close to the best one. For example, the best classification accuracy for *Spectheart* and *LFW* data set are 0.7852 and 0.6525, respectively, and the SC-GMM classifier obtains 0.7778 and 0.6450 (only 0.0074 and 0.0075 inferior to the best).

These observations show that SC-GMM is a competitive classifier.

Table 6.6: The classification accuracies of *k*NN, Linear-SVM, RBF-SVM, DT, NB and SC-GMM on all data sets

Methods / Data sets	KNN	Linear-SVM	RBF-SVM	DT	NB	SC-GMM
Climate	0.8981 ± 0.0363	0.9130 ± 0.0389	0.9130 ± 0.0389	0.9093 ± 0.0326	**0.9519 ± 0.0301**	0.9259 ± 0.0381
Dermatology	0.8892 ± 0.0560	**0.9730 ± 0.0242**	0.2725 ± 0.0564	0.9162 ± 0.0282	0.8946 ± 0.0490	0.9622 ± 0.0573
Glass1	**0.8364 ± 0.0739**	0.6227 ± 0.0933	0.7636 ± 0.0881	0.7182 ± 0.0755	0.5545 ± 0.1282	0.7955 ± 0.0996
Ionosphere	0.8556 ± 0.0593	0.8583 ± 0.0548	0.8972 ± 0.0449	0.8806 ± 0.0482	0.9000 ± 0.0356	**0.9250 ± 0.0330**
M1-C1	0.8500 ± 0.0500	0.8333 ± 0.0475	0.5604 ± 0.0776	0.7479 ± 0.0529	0.7500 ± 0.0632	**0.8854 ± 0.1111**
Seeds	0.9238 ± 0.0774	0.9286 ± 0.0488	0.9095 ± 0.0688	0.9190 ± 0.0879	0.8857 ± 0.0632	**0.9333 ± 0.0497**
Sonar	0.8381 ± 0.0830	0.5952 ± 0.1171	**0.8857 ± 0.0646**	0.7429 ± 0.0639	0.6905 ± 0.0857	0.8619 ± 0.1187
Spectheart	0.7519 ± 0.0723	0.7296 ± 0.0795	**0.7852 ± 0.0637**	0.7370 ± 0.0811	0.6704 ± 0.0852	0.7778 ± 0.0637
Hayes-roth	0.7214 ± 0.1479	0.4571 ± 0.0969	0.7429 ± 0.01286	0.6357 ± 0.0811	0.6429 ± 0.0811	**0.7643 ± 0.1429**
MF-fou	0.8210 ± 0.0156	0.6490 ± 0.0303	**0.8445 ± 0.0188**	0.6045 ± 0.0252	0.7505 ± 0.0217	0.8150 ± 0.0328
LFW	**0.6525 ± 0.0656**	0.5650 ± 0.0930	0.4500 ± 0.0354	0.5950 ± 0.0731	0.6300 ± 0.0696	0.6450 ± 0.0678

6.5 Summary

In this chapter we propose a novel GMM classifier, called SC-GMM, which quickly finds the optimal number of Gaussian components for each class, and classifies a new sample by utilising these Gaussian components of each class. Extensive experiments show that SC-GMM significantly improves the classification performance of the original GMM classifier. Compared with three variants of the original GMM classifier, namely, AIC-GMM, BIC-GMM and VBGM, SC-GMM is more efficient since it not only finds the optimal number of Gaussian components for each class far more quickly than AIC-GMM and BIC-GMM, but also can achieve classification accuracy that is competitive with them. In addition, compared with five other widely used classifiers, SC-GMM enables the original GMM classifier to become competitive with them.

CHAPTER VII

Conclusions and Future Work

This chapter concludes the thesis with a summary of the studies and contributions, and an outlook for future work.

7.1 Conclusions

In this thesis, we focus on cluster-based supervised classification with the aim of developing cluster-based feature extraction methods and classifiers to improve the performance of supervised classification. Details of our contributions are summarised in the following:

Chapter III conducted a study on the importance of considering within-class multimodality in the process of feature extraction. In our real-world classification problems there exist multiple subclasses (or clusters) within a class; in other words, the underlying data distribution is within-class multimodal. One example is face recognition, where a face (i.e. a class) may be presented in frontal view or side view, corresponding to different modalities. But, how to address the within-class multimodality issue is still an unsolved problem. To resolve this problem, this study is guided by five key questions, and conducted through experimentation on artificial data and real data. In addition, we established a case for within-class multimodal classification that is characterised by the concurrent maximisation of between-class separation, between-subclass separation and

within-class compactness. The extensive experimental results led to the following useful findings: 1) when within-class multimodality is present, the concurrent maximisation of between-class separation, between-subclass separation and within-class compactness can lead to significant performance gains; 2) within-class multimodal classification offers a competitive solution to face recognition under different lighting and face pose conditions, where each lighting/pose condition corresponds to a separate modality in the data space; 3) there is a correlation between multimodality and performance gain in within-class multimodality classification. Optimal performance can be expected if the number of modalities in the within-class multimodality classification algorithm is the same as the true number of within-class modalities.

Chapter IV proposed a new cluster-based feature extraction method, which is called global subclass discriminant analysis (GSDA). It is well known that linear discriminant analysis (LDA) provides an analytic solution to the mathematical problem of finding a subspace that maximises interclass distance and concurrently minimises intraclass distance. It works very well on linearly separable data, but not well on data that are not linearly separable. A number of variants have been proposed in the literature to address the nonlinear data problem, including subclass-based LDAs. An important problem with these variants is that subclasses are selected in a local, class-specific way, and the optimisation objective becomes complex and thus the meaning of the optimisation process is less clear. To solve this problem, GSDA is developed. GSDA selects subclasses in a *global way* by clustering the whole data set, rather than one class at a time, and derives class-specific subclasses on the basis of these *global subclasses*. It then seeks to maximise interclass distance and minimise intraclass distance based on these class-specific subclasses, which means that it is optimally trying to separate all subclasses and concurrently to make them as compact as possible. Clustering is done using an algorithm called *rough-refine clustering*, which is proposed in this thesis. GSDA is extensively evaluated on a wide range of data through comparison with the state-of-the-art in LDA

algorithms, including subclass-based LDA variants such as SDA, MSDA and SSDA, as well as kernel-based nonlinear LDA variants, including KDA, KMSDA and KSDA. Experimental results demonstrated clear advantages of GSDA over these benchmarks in terms of accuracy and run time.

Chapter V presented the cluster-based data relabelling (CBDR) method to enable linear classifiers to work on nonlinear data. It is well known that linear classifiers are generally simpler and more explainable than their nonlinear variants, and they can achieve satisfactory classification performance on linearly separable data. However, not surprisingly, performance can be unsatisfactory on data that is not linearly separable, or *nonlinear data* for short. Linear classifiers can be extended to deal with nonlinear data, and one common approach is to introduce kernels into linear classifiers. However, this has to be done on a case-by-case basis. In contrast, CBDR is a generic method. CBDR can be easily applied to different linear classifiers. To achieve this, CBDR partitions the data set into several nonoverlapping class-specific clusters, relabels each cluster separately, and then applies a linear classifier to the relabelled data to seek cluster-based linear decision boundaries instead of class-based decision boundaries. CBDR has been evaluated along with three classic linear classifiers: linear discriminant analysis classifier (LDAC), linear support vector machine (LSVM), and linear multilayer perceptron (LMLP). Experimental results using a large number of real data sets and artificial data sets demonstrated the utility of CBDR, which significantly enhanced the classification performance of these three linear classifiers. Additionally, we compared some CBDR-equipped nonlinear classifiers, naive Bayes classifier and decision tree, with these classifiers alone by experiments. Extensive experimental results show that the CBDR-equipped classifiers are always better than or comparable to these classifiers alone in classification performance. Furthermore, the most significant outperformance is observed on imbalanced data in both cases.

Chapter VI proposed a novel Gaussian mixture model (GMM), called *SC-GMM*

classifier, where SC denotes separability criterion. GMM is usually used for unsupervised learning to learn the subpopulations and the subpopulation assignment automatically. It is also used for supervised learning or classification to learn the boundary of subpopulations. However, the performance of GMM as a classifier is not impressive compared with other conventional classifiers such as k-nearest neighbours, support vector machine, decision tree and naive Bayes. To enhance the classification performance of GMM on the supervised classification problem, in this chapter we developed the SC-GMM classifier. The SC-GMM classifier finds the optimal number of Gaussian components for each class based on the separability criterion and then determines the parameters of these Gaussian components by using the Expectation Maximisation algorithm. Extensive experiments have been carried out on classification tasks from general data mining to face verification. Results show that SC-GMM significantly outperforms the original GMM classifier. In addition, experimental results also show that SC-GMM is comparable in classification accuracy to three variants of GMM classifier: Akaike information criterion-based GMM (AIC-GMM), Bayesian information criterion-based GMM (BIC-GMM), and variational Bayesian Gaussian mixture (VBGM). However, SC-GMM is significantly more efficient than both AIC-GMM and BIC-GMM. Furthermore, compared with k-nearest neighbours, support vector machine, decision tree and naive Bayes, SC-GMM classifier achieves competitive classification performance.

7.2 Future Work

The research presented in this thesis focuses mainly on general classification tasks. However, the idea of applying clustering to classification has huge potential in many specific scenarios. Image-based face recognition is one such scenario, as face data is structured. With an appropriate feature extraction method, the face images of a person tend to form multiple clusters according to properties of the faces, for example, age and pose. Future work will study how to apply cluster-based classification (CBC) to

age-related face recognition (ARFR) and pose-related face recognition (PRFR).

In ARFR, we will focus on two aspects: age-invariant face recognition (AIFR) and age estimation in face recognition (AEFR). Facial ageing affects the shape and texture of the face and thus the accuracy of face recognition. AIFR refers to an automatic face recognition technology that can recognise faces despite the appearance variations on age. AIFR methods are very important in some applications, including identifying missing children, law enforcement, and multiple registration detection. The existing AIFR techniques include generative approaches, discriminative methods and deep learning-based methods [103]. Generative approaches rely on age normalisation before matching, namely transforming the testing face image to the same age as the gallery face image. Discriminative methods mainly refer to the age invariant feature descriptors, which encode the visual clues in face images. In recent years, deep learning-based methods, especially Convolutional Neural Networks (CNN), have emerged as a powerful machine learning model. Experimental results show that CNN outperform the generative and discriminative methods. In our future work, we will use the deep learning method to extract the features of the face images but will integrate the CBC idea, so as to perform more local optimisation of the data space in terms of age to achieve better recognition performance. In AEFR, we will attempt to establish the classification boundary between the clusters, so that face images can be classified and sorted according to age while recognising the identity of the face.

In PRFR, we will focus on two aspects: pose-invariant face recognition (PIFR) and pose estimation in face recognition (PEFR). PIFR is the process of recognising people using facial images captured in arbitrary poses, which is of great application value in some scenarios like train stations, airports and banks. The subject's attention is rarely focused on the surveillance cameras, so the camera in these public places usually capture only non-frontal face images. When the non-frontal face images are used to match the frontal face images in the gallery, the accuracy of face recognition is typically greatly reduced.

Therefore, PIFR methods are needed to solve this problem. The existing PIFR methods can be divided into four categories [26]: (a) pose transformation for face matching; (b) finding a shared subspace of different poses; (c) pose-invariant features; (d) hybrid methods of the above strategies. With CBC, we will try to obtain the clusters of different poses. Once the pose clusters are obtained, the pose-invariant features can be extracted by local and global data space optimisation. In PEFR, we will attempt to establish the classification boundary between clusters, so that the face images can be classified and sorted according to pose while recognising the identity of the face.

REFERENCES

[1] Hirotugu Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974.

[2] Jesús Alcalá-Fdez, Alberto Fernández, Julián Luengo, Joaquín Derrac, Salvador García, Luciano Sánchez, and Francisco Herrera. Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic & Soft Computing*, 17:255–287, 2011.

[3] Jesús Alcalá-Fdez, Luciano Sanchez, Salvador Garcia, Maria Jose del Jesus, Sebastian Ventura, Josep Maria Garrell, José Otero, Cristóbal Romero, Jaume Bacardit, Victor M. Rivas, J. C. Fernández, and F. Herrera. Keel: a software tool to assess evolutionary algorithms for data mining problems. *Soft Computing*, 13(3):307–318, 2009.

[4] Mohammad Alhawarat and Ahmad O Aseeri. A superior arabic text categorization deep model (satcdm). *IEEE Access*, 8:24653–24661, 2020.

[5] Hagai Attias. A variational baysian framework for graphical models. In *Advances in Neural Information Processing Systems*, pages 209–215, 2000.

[6] Gaston Baudat and Fatiha Anouar. Generalized discriminant analysis using a kernel approach. *Neural Computation*, 12(10):2385–2404, 2000.

[7] Peter N. Belhumeur, João P Hespanha, and David J. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):711–720, 1997.

[8] Christopher M. Bishop. *Neural networks for pattern recognition*. Oxford university press, 1995.

[9] David M. Blei and Michael I. Jordan. Variational inference for dirichlet process mixtures. *Bayesian Analysis*, 1(1):121–143, 2006.

[10] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 144–152, 1992.

[11] Athman Bouguettaya, Qi Yu, Xumin Liu, Xiangmin Zhou, and Andy Song. Efficient agglomerative hierarchical clustering. *Expert Systems with Applications*, 42(5):2785–2797, 2015.

[12] Anissa Bouzalmat, Jamal Kharroubi, and Arsalane Zarghili. Comparative study of pca, ica, lda using svm classifier. *Journal of Emerging Technologies in Web Intelligence*, 6(1):64–68, 2014.

[13] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[14] Christopher JC Burges, Bernhard Scholkopf, and Alexander J Smola. *Advances in kernel methods: support vector learning*. MIT press Cambridge, MA, USA:, 1999.

[15] Jean-François Cardoso and Antoine Souloumiac. Blind beamforming for non-gaussian signals. In *IEE proceedings F (radar and signal processing)*, volume 140, pages 362–370. IET, 1993.

[16] Li-Fen Chen, Hong-Yuan Mark Liao, Ming-Tat Ko, Ja-Chen Lin, and Gwo-Jong Yu. A new lda-based face recognition system which can solve the small sample size problem. *Pattern recognition*, 33(10):1713–1726, 2000.

[17] Kateryna Chumachenko, Jenni Raitoharju, Moncef Gabbouj, and Alexandros Iosifidis. Incremental fast subclass discriminant analysis. *arXiv preprint arXiv:2002.04348*, 2020.

[18] Ronan Collobert and Samy Bengio. Links between perceptrons, mlps and svms. In *Proceedings of the Twenty-first International Conference on Machine Learning*, page 23, 2004.

[19] Pierre Comon. Independent component analysis, a new concept? *Signal Processing*, 36(3):287–314, 1994.

[20] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[21] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[22] Lieven De Lathauwer, Josphine Castaing, and Jean-Franois Cardoso. Fourth-order cumulant-based blind identification of underdetermined mixtures. *IEEE Transactions on Signal Processing*, 55(6):2965–2973, 2007.

[23] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

[24] Zhiwei Deng, Arash Vahdat, Hexiang Hu, and Greg Mori. Structure inference machines: Recurrent neural networks for analyzing relations in group activity recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4772–4781, 2016.

[25] Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository, 2017.

[26] Changxing Ding and Dacheng Tao. A comprehensive survey on pose-invariant face recognition. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 7(3):1–42, 2016.

[27] Shifei Ding, Hong Zhu, Weikuan Jia, and Chunyang Su. A survey on feature extraction for pattern recognition. *Artificial Intelligence Review*, 37(3):169–180, 2012.

[28] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[29] Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification*. John Wiley & Sons, 2012.

[30] Joseph C Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3:32–57, 1973.

[31] Yan Em, Feng Gag, Yihang Lou, Shiqi Wang, Tiejun Huang, and Ling-Yu Duan. Incorporating intra-class variance to fine-grained visual recognition. In *2017 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1452–1457, July 2017.

[32] Mario A. T. Figueiredo and Anil K. Jain. Unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):381–396, 2002.

[33] Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188, 1936.

[34] Ronald A Fisher. The statistical utilization of multiple measurements. *Annals of Eugenics*, 8(4):376–386, 1938.

[35] Jerome H Friedman. Regularized discriminant analysis. *Journal of the American statistical association*, 84(405):165–175, 1989.

[36] Zhouyu Fu, Antonio Robles-Kelly, and Jun Zhou. Mixing linear svms for nonlinear classification. *IEEE Transactions on Neural Networks*, 21(12):1963–1975, 2010.

[37] Shiming Ge, Shengwei Zhao, Chenyu Li, Yu Zhang, and Jia Li. Efficient low-resolution face recognition via bridge distillation. *IEEE Transactions on Image Processing*, 2020.

[38] Nikolaos Gkalelis, Vasileios Mezaris, and Ioannis Kompatsiaris. Mixture subclass discriminant analysis. *IEEE Signal Processing Letters*, 18(5):319–322, 2011.

[39] Nikolaos Gkalelis, Vasileios Mezaris, Ioannis Kompatsiaris, and Tania Stathaki. Mixture subclass discriminant analysis link to restricted gaussian model and other generalizations. *IEEE Transactions on Neural Networks and Learning Systems*, 24(1):8–21, 2012.

[40] JJ Glen. Mathematical programming models for piecewise-linear discriminant analysis. *Journal of the Operational Research Society*, 56(3):331–341, 2005.

[41] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[42] Yanming Guo, Yu Liu, Ard Oerlemans, Songyang Lao, Song Wu, and Michael S Lew. Deep learning for visual understanding: A review. *Neurocomputing*, 187:27–48, 2016.

[43] David J Hand and Veronica Vinciotti. Local versus global models for classification problems: fitting models where it matters. *The American Statistician*, 57(2):124–131, 2003.

[44] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

[45] Fumio Hayashi. *Econometrics*. Princeton University Press, 2000.

[46] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[47] Marko Heikkilä, Matti Pietikäinen, and Cordelia Schmid. Description of interest regions with center-symmetric local binary patterns. In *Computer Vision, Graphics and Image Processing*, pages 58–69. Springer, 2006.

[48] Tin Kam Ho and Mitra Basu. Complexity measures of supervised classification problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):289–300, 2002.

[49] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

[50] Meiyan Huang, Wei Yang, Qianjin Feng, and Wufan Chen. Longitudinal measurement and hierarchical classification framework for the prediction of alzheimer's disease. *Scientific Reports*, 7(1):1–13, 2017.

[51] Aapo Hyvarinen. Fast and robust fixed-point algorithms for independent component analysis. *IEEE transactions on Neural Networks*, 10(3):626–634, 1999.

[52] Aapo Hyvärinen and Erkki Oja. Independent component analysis: algorithms and applications. *Neural Networks*, 13(4-5):411–430, 2000.

[53] Alexandros Iosifidis and Moncef Gabbouj. Scaling up class-specific kernel discriminant analysis for large-scale face verification. *IEEE Transactions on Information Forensics and Security*, 11(11):2453–2465, 2016.

[54] Finn V Jensen. *An introduction to Bayesian networks*, volume 210. UCL press London, 1996.

[55] Yanli Ji, Yang Yang, Fumin Shen, Heng Tao Shen, and Wei-Shi Zheng. Arbitrary-view human action recognition: A varying-view rgb-d action dataset. *IEEE Transactions on Circuits and Systems for Video Technology*, 2020.

[56] Wenjuan Jia, Yingjie Deng, Chenyang Xin, Xiaodong Liu, and Witold Pedrycz. A classification algorithm with linear discriminant analysis and axiomatic fuzzy sets. *Mathematical Foundations of Computing*, 2(1):73–81, 2019.

[57] Parunyou Julayanont, Nikolaus R McFarland, and Kenneth M Heilman. Mild cognitive impairment and dementia in motor manifest huntington's disease: Classification and prevalence. *Journal of the Neurological Sciences*, 408:116523, 2020.

[58] Meina Kan, Dong Xu, Shiguang Shan, Wen Li, and Xilin Chen. Learning prototype hyperplanes for face verification in the wild. *IEEE Transactions on Image Processing*, 22(8):3310–3316, 2013.

[59] Cheolmin Kim and Diego Klabjan. A simple and fast algorithm for l1-norm kernel pca. *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[60] Jaeyoung Kim, Sion Jang, Eunjeong Park, and Sungchul Choi. Text classification using capsules. *Neurocomputing*, 376:214–221, 2020.

[61] Stefan Knerr, Léon Personnaz, and Gérard Dreyfus. Single-layer learning revisited: a stepwise procedure for building and training a neural network. In *Neurocomputing*, pages 41–50. Springer, 1990.

[62] Ulrich H-G Kreßel, B Schölkopf, CJC Burges, and AJ Smola. Pairwise classification and support vector machines, advances in kernel methods: support vector learning, 1999.

[63] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[64] Harold W Kuhn and Albert W Tucker. Nonlinear programming. In *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, pages 481–492. University of California Press, 1951.

[65] Derrick Norman Lawley and Albert Ernest Maxwell. Factor analysis as a statistical method. *Journal of the Royal Statistical Society. Series D (The Statistician)*, 12(3):209–229, 1962.

[66] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[67] Michael S Lewicki and Terrence J Sejnowski. Learning overcomplete representations. *Neural computation*, 12(2):337–365, 2000.

[68] Chunna Li, Yuanhai Shao, Wotao Yin, and Mingzeng Liu. Robust and sparse linear discriminant analysis via an alternating direction method of multipliers. *IEEE Transactions on Neural Networks and Learning Systems*, 2019.

[69] Duanshun Li, Anran Cong, and Shuai Guo. Sewer damage detection from imbalanced cctv inspection data using deep convolutional neural networks with hierarchical classification. *Automation in Construction*, 101:199–208, 2019.

[70] Huaxiong Li, Libo Zhang, Bing Huang, and Xianzhong Zhou. Cost-sensitive dual-bidirectional linear discriminant analysis. *Information Sciences*, 510:283–303, 2020.

[71] Jinyan Li, Simon Fong, Yan Zhuang, and Richard Khoury. Hierarchical classification in text mining for sentiment analysis of online news. *Soft Computing*, 20(9):3411–3420, 2016.

[72] Jonathan Q Li and Andrew R Barron. Mixture density estimation. In *Advances in Neural Information Processing Systems*, pages 279–285, 2000.

[73] Zhifeng Li, Dahua Lin, and Xiaoou Tang. Nonparametric discriminant analysis for face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(4):755–761, 2009.

[74] Moshe Lichman. UCI machine learning repository, 2013.

[75] Gang Liu and Jiabao Guo. Bidirectional lstm with attention mechanism and convolutional layer for text classification. *Neurocomputing*, 337:325–338, 2019.

[76] Quande Liu, Lequan Yu, Luyang Luo, Qi Dou, and Pheng Ann Heng. Semi-supervised medical image classification with relation-driven self-ensembling model. *IEEE Transactions on Medical Imaging*, 2020.

[77] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. Sphereface: Deep hypersphere embedding for face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 212–220, 2017.

[78] Weiyin Loh. Classification and regression trees. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1):14–23, 2011.

[79] Gilles Louppe. Understanding random forests: From theory to practice. *arXiv preprint arXiv:1407.7502*, 2014.

[80] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[81] Dwi Sianto Mansjur and Biing Hwang Juang. Incremental learning of mixture models for simultaneous estimation of class distribution and inter-class decision boundaries. In *2008 19th International Conference on Pattern Recognition*, pages 1–4. IEEE, 2008.

[82] Aleix M Martínez and Avinash C Kak. Pca versus lda. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(2):228–233, 2001.

[83] Aleix M Martinez and Manli Zhu. Where are linear feature extraction methods applicable? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(12):1934–1944, 2005.

[84] MATLAB. *9.1.0.441655 (R2016b)*. The MathWorks Inc., Natick, Massachusetts, 2016.

[85] Geoffrey J McLachlan. *Discriminant analysis and statistical pattern recognition*, volume 544. John Wiley & Sons, 2004.

[86] Sebastian Mika, Gunnar Ratsch, Jason Weston, Bernhard Scholkopf, and Klaus-Robert Mullers. Fisher discriminant analysis with kernels. In *Neural Networks for Signal Processing IX: Proceedings of the 1999 IEEE Signal Processing Society Workshop*, pages 41–48. IEEE, 1999.

[87] Sebastian Mika, Bernhard Schölkopf, Alex J Smola, Klaus-Robert Müller, Matthias Scholz, and Gunnar Rätsch. Kernel pca and de-noising in feature spaces. In *Advances in neural information processing systems*, pages 536–542, 1999.

[88] Andrew W Moore. Very fast em-based mixture model clustering using multiresolution kd-trees. In *Advances in Neural Information Processing Systems*, pages 543–549, 1999.

[89] Alejandro Moreo, Andrea Esuli, and Fabrizio Sebastiani. Learning to weight for text classification. *IEEE Transactions on Knowledge and Data Engineering, Forthcoming*, 2020.

[90] David Murray, Lina Stankovic, Vladimir Stankovic, Srdjan Lulic, and Srdjan Sladojevic. Transferability of neural network approaches for low-rate energy disaggregation. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8330–8334. IEEE, 2019.

[91] Fionn Murtagh and Pedro Contreras. Algorithms for hierarchical clustering: an overview, ii. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 7(6):e1219, 2017.

[92] Andrew Y Ng and Michael I Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in Neural Information Processing Systems*, pages 841–848, 2002.

[93] Feiping Nie, Zheng Wang, Rong Wang, Zhen Wang, and Xuelong Li. Adaptive local linear discriminant analysis. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 14(1):1–19, 2020.

[94] Kai Niu, Yan Huang, Wanli Ouyang, and Liang Wang. Improving description-based person re-identification by multi-granularity image-text alignments. *IEEE Transactions on Image Processing*, 29:5542–5556, 2020.

[95] Shaoning Pang, Seiichi Ozawa, and Nikola Kasabov. Incremental linear discriminant analysis for classification of data streams. *IEEE transactions on Systems, Man, and Cybernetics, part B (Cybernetics)*, 35(5):905–914, 2005.

[96] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[97] Elżbieta Pekalska and Bernard Haasdonk. Kernel discriminant analysis for positive definite and indefinite kernels. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(6):1017–1032, 2008.

[98] P Jonathon Phillips, Hyeonjoon Moon, Syed A Rizvi, and Patrick J Rauss. The feret evaluation methodology for face-recognition algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(10):1090–1104, 2000.

[99] Anastasia Podosinnikova, Amelia Perry, Alexander Wein, Francis Bach, Alexandre d'Aspremont, and David Sontag. Overcomplete independent component analysis via sdp. *arXiv preprint arXiv:1901.08334*, 2019.

[100] Lawrence Rabiner and B Juang. An introduction to hidden markov models. *IEEE ASSP Magazine*, 3(1):4–16, 1986.

[101] C Radhakrishna Rao. The utilization of multiple measurements in problems of biological classification. *Journal of the Royal Statistical Society. Series B (Methodological)*, 10(2):159–203, 1948.

[102] Michael A Sartori and Panos J Antsaklis. A simple method to derive bounds on the size and to train multilayer neural networks. *IEEE Transactions on Neural Networks*, 2(4):467–471, 1991.

[103] Manisha M Sawant and Kishor M Bhurchandi. Age invariant face recognition: a survey on facial aging databases, techniques and effect of aging. *Artificial Intelligence Review*, 52(2):981–1008, 2019.

[104] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

[105] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823, 2015.

[106] Gideon Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

[107] George AF Seber and Alan J Lee. *Linear regression analysis*, volume 329. John Wiley & Sons, 2012.

[108] Alok Sharma and Kuldip K Paliwal. Linear discriminant analysis for the small sample size problem: an overview. *International Journal of Machine Learning and Cybernetics*, 6(3):443–454, 2015.

[109] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[110] Harinder P Singh, Coryn AL Bailer-Jones, and Ranjan Gupta. Introduction to artificial neural networks. 2001.

[111] Rosanna Soentpiet. *Advances in kernel methods: support vector learning*. MIT press, 1999.

[112] Abdulhamit Subasi and M Ismail Gursoy. Eeg signal classification using pca, ica, lda and support vector machines. *Expert Systems with Applications*, 37(12):8659–8666, 2010.

[113] Zhongxi Sun, Jun Li, and Changyin Sun. Kernel inverse fisher discriminant analysis for face recognition. *Neurocomputing*, 134:46–52, 2014.

[114] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[115] Shin'ichi Tamura and Masahiko Tateishi. Capabilities of a four-layered feedforward neural network: four layers versus three. *IEEE Transactions on Neural Networks*, 8(2):251–255, 1997.

[116] Yee Whye Teh, Max Welling, Simon Osindero, and Geoffrey E Hinton. Energy-based models for sparse overcomplete representations. *Journal of Machine Learning Research*, 4(Dec):1235–1260, 2003.

[117] Alaa Tharwat. Linear vs. quadratic discriminant analysis classifier: a tutorial. *International Journal of Applied Pattern Recognition*, 3(2):145–180, 2016.

[118] Virendra P Vishwakarma and Sahil Dalal. A novel non-linear modifier for adaptive illumination normalization for robust face recognition. *Multimedia Tools and Applications*, pages 1–27, 2020.

[119] Huan Wan, Hui Wang, Gongde Guo, and Xin Wei. Separability-oriented subclass discriminant analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(2):409–422, 2018.

[120] Fei Wang, Quan Wang, Feiping Nie, Zhongheng Li, Weizhong Yu, and Rong Wang. Unsupervised linear discriminant analysis for jointly clustering and subspace learning. *IEEE Transactions on Knowledge and Data Engineering*, 2019.

[121] Xin Wei, Hui Wang, Bryan Scotney, and Huan Wan. Minimum margin loss for deep face recognition. *Pattern Recognition*, 97:107012, 2020.

[122] Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. A discriminative feature learning approach for deep face recognition. In *European Conference on Computer Vision*, pages 499–515. Springer, 2016.

[123] Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. A comprehensive study on center loss for deep face recognition. *International Journal of Computer Vision*, 127(6-7):668–683, 2019.

[124] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and Intelligent laboratory Systems*, 2(1-3):37–52, 1987.

[125] Lior Wolf, Tal Hassner, and Itay Maoz. Face recognition in unconstrained videos with matched background similarity. In *2011 Computer Vision and Pattern Recognition (CVPR)*, pages 529–534. IEEE, 2011.

[126] Lior Wolf, Tal Hassner, and Yaniv Taigman. Similarity scores based on background samples. In *Asian Conference on Computer Vision*, pages 88–97. Springer, 2009.

[127] Baile Xu, Shaofeng Shen, Furao Shen, and Jian Zhao. Locally linear svms based on boundary anchor points encoding. *Neural Networks*, 117:274–284, 2019.

[128] Yuan Xue, Tao Xu, Han Zhang, L Rodney Long, and Xiaolei Huang. Segan: Adversarial network with multi-scale l 1 loss for medical image segmentation. *Neuroinformatics*, 16(3-4):383–392, 2018.

[129] Hao Yang, Chunfeng Yuan, Li Zhang, Yunda Sun, Weiming Hu, and Stephen J Maybank. Sta-cnn: convolutional spatial-temporal attention learning for action recognition. *IEEE Transactions on Image Processing*, 2020.

[130] Jieping Ye, Qi Li, Hui Xiong, Haesun Park, Ravi Janardan, and Vipin Kumar. Idr/qr: An incremental dimension reduction algorithm via qr decomposition. *IEEE Transactions on Knowledge and Data Engineering*, 17(9):1208–1222, 2005.

[131] Di You, Onur C Hamsici, and Aleix M Martinez. Kernel optimization in discriminant analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(3):631–638, 2010.

[132] Feifei Zhang, Tianzhu Zhang, Qirong Mao, and Changsheng Xu. Geometry guided pose-invariant facial expression recognition. *IEEE Transactions on Image Processing*, 29:4445–4460, 2020.

[133] Harry Zhang. The optimality of naive bayes, flairs conference, 2004.

[134] Pengfei Zhang, Cuiling Lan, Wenjun Zeng, Junliang Xing, Jianru Xue, and Nanning Zheng. Semantics-guided neural networks for efficient skeleton-based human action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1112–1121, 2020.

[135] Wuming Zhang, Xi Zhao, Jean-Marie Morvan, and Liming Chen. Improving shadow suppression for illumination robust face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP(99):1–1, 2018.

[136] Bo Zhou, Benhui Chen, and Jinglu Hu. Quasi-linear support vector machine for nonlinear classification. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 97(7):1587–1594, 2014.

[137] Manli Zhu and Aleix M Martinez. Subclass discriminant analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(8):1274–1286, 2006.